Medical Statistics with R

Dr. Gulser Caliskan
Prof. Giuseppe Verlato

Unit of Epidemiology and Medical Statistics
Department of Diagnostics and Public Health
University of Verona, Italy

INDEX OF LESSON 1

What Is R?

R is a language and environment for statistical computing and graphics.

R provides a wide variety of statistical (linear and nonlinear modelling, classical statistical tests, time-series analysis, classification, clustering, …) and graphical techniques, and is highly extensible.

The S language is often the vehicle of choice for research in statistical methodology, and R provides an Open Source route to participation in that activity.

How We Can Install R?

Download R-4.0.4 for Windows

cran.r-project.org/bin/windows/base/

R-4.0.4 for Windows (32/64 bit)

Download R 4.0.4 for Windows (85 megabytes, 32/64 bit)

Installation and other instructions
New features in this version

If you want to double-check that the package you have downloaded matches the package distributed by CRAN, you can compare the md5sum of the .exe to the fingerprint on the master server. You will need a version of md5sum for windows: both graphical and command line versions are available.

Frequently asked questions

- Does R run under my version of Windows?
- How do I update packages in my previous version of R?
- Should I run 32-bit or 64-bit R?

Please see the R FAQ for general information about R and the R Windows FAQ for Windows-specific information.

Other builds

- Patches to this release are incorporated in the r-patched snapshot build.
- A build of the development version (which will eventually become the next major release of R) is available in the r-devel snapshot build.
- Previous releases

Note to webmasters: A stable link which will redirect to the current Windows binary release is <CRAN MIRROR>/bin/windows/base/release.html.

Last change: 2021-02-15

Type here to search

ENG IT 10:06 PM 3/16/2021

cran.r-project.org/bin/windows/base/

R-4.0.4 for Windows (32/64 bit)

Download R 4.0.4 for Windows (85 megabytes, 32/64 bit)

Installation and other instructions
New features in this version

If you want to double-check that the package you have downloaded matches the package distributed by CRAN, you can compare the md5sum of the .exe to the fingerprint o
and command line versions are available.

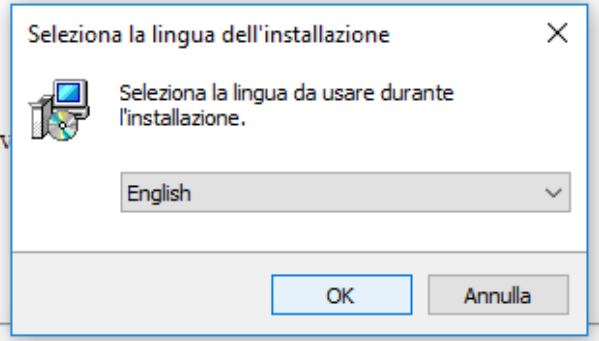
Frequently asked questions

- Does R run under my version of Windows?
- How do I update packages in my previous version of R?
- Should I run 32-bit or 64-bit R?

Please see the R FAQ for general information about R and the R Windows FAQ for Windows-specific information.

- Patches to this release are incorporated in the r-patched snapshot build.
- A build of the development version (which will eventually become the next major release of R) is av
- Previous releases

Note to webmasters: A stable link which will redirect to the current Windows binary release is
<CRAN MIRROR>/bin/windows/base/release.html.



Last change: 2021-02-15

cran.r-project.org/bin/windows/base/

R-4.0.4 for Windows (32/64 bit)

Download R 4.0.4 for Windows (85 megabytes, 32/64 bit)

Installation and other instructions
New features in this version

If you want to double-check that the package you have downloaded matches t the
and command line versions are available.

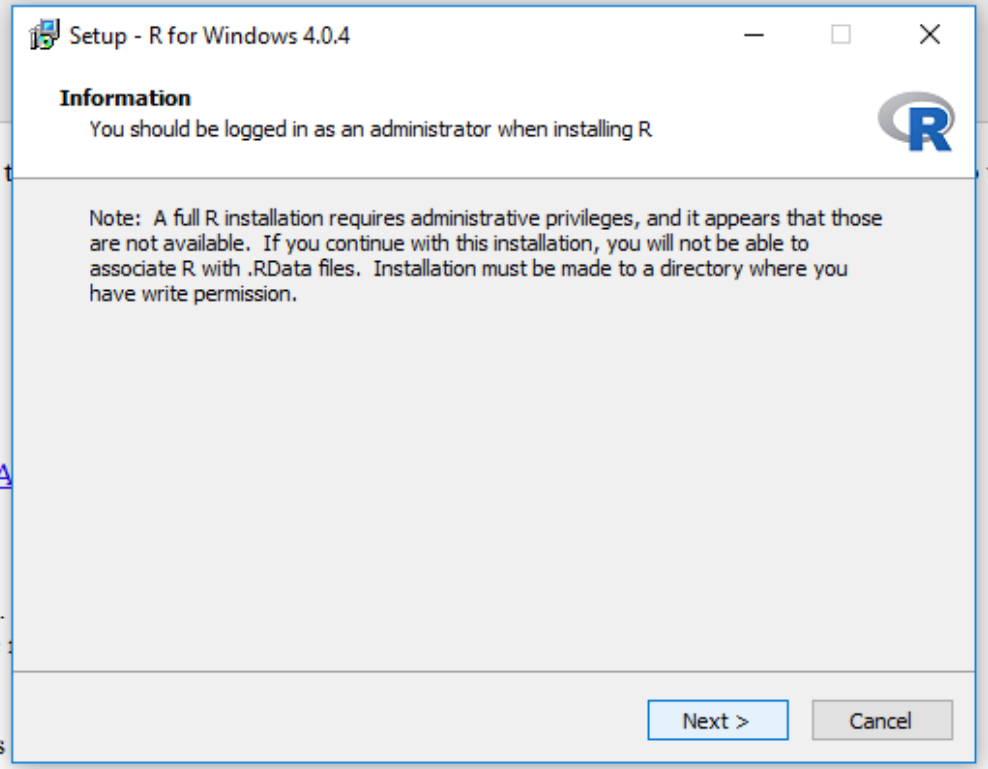
- Does R run under my version of Windows?
- How do I update packages in my previous version of R?
- Should I run 32-bit or 64-bit R?

Please see the R FAQ for general information about R and the R Windows FA

- Patches to this release are incorporated in the r-patched snapshot build.
- A build of the development version (which will eventually become the
- Previous releases

Note to webmasters: A stable link which will redirect to the current Windows
<CRAN MIRROR>/bin/windows/base/release.html.

Last change: 2021-02-15



R-4.0.4 for Windows (32/64 bit)

Download R 4.0.4 for Windows (85 megabytes, 32/64 bit)

Installation and other instructions
New features in this version

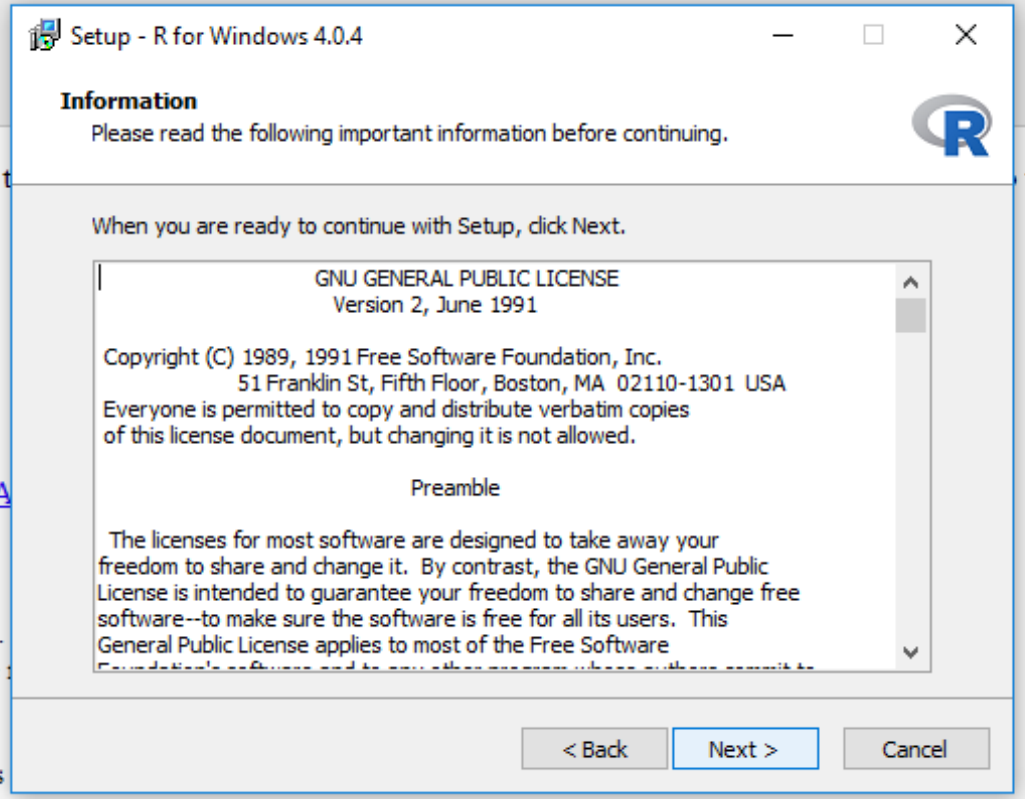
If you want to double-check that the package you have downloaded matches t the fingerprint on t
and command line versions are available.

- Does R run under my version of Windows?
- How do I update packages in my previous version of R?
- Should I run 32-bit or 64-bit R?

Please see the R FAQ for general information about R and the R Windows FA

- Patches to this release are incorporated in the r-patched snapshot build.
- A build of the development version (which will eventually become the
- Previous releases

Note to webmasters: A stable link which will redirect to the current Windows
<CRAN MIRROR>/bin/windows/base/release.html.



Last change: 2021-02-15

cran.r-project.org/bin/windows/base/

R-4.0.4 for Windows (32/64 bit)

Download R 4.0.4 for Windows (85 megabytes, 32/64 bit)

Installation and other instructions
New features in this version

If you want to double-check that the package you have downloaded matches t the f
and command line versions are available.

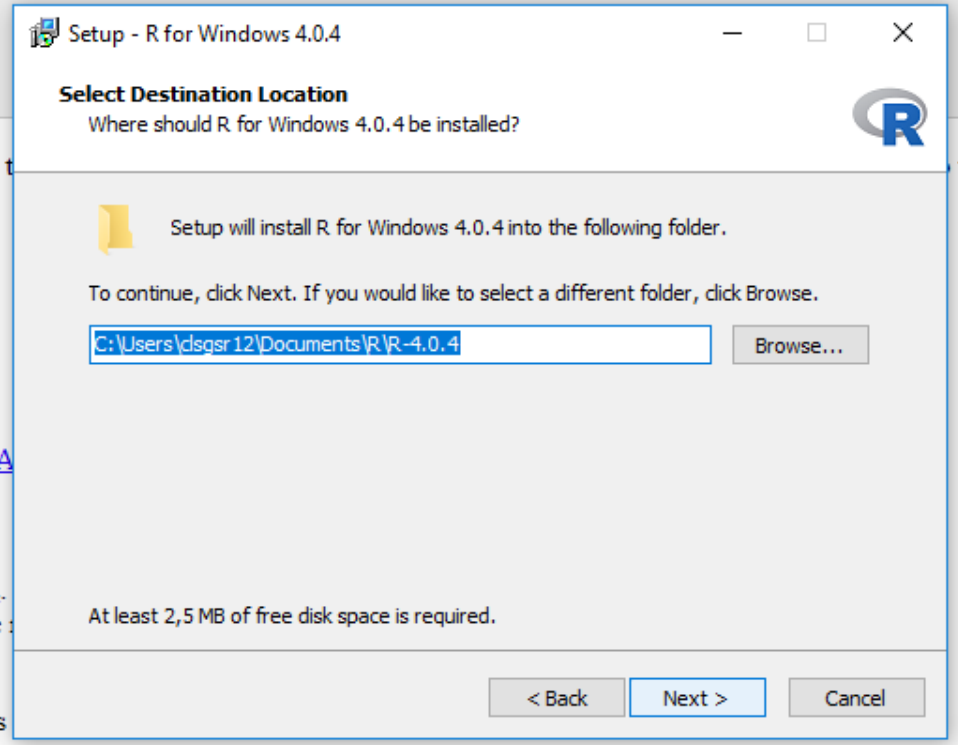
- Does R run under my version of Windows?
- How do I update packages in my previous version of R?
- Should I run 32-bit or 64-bit R?

Please see the R FAQ for general information about R and the R Windows FA

- Patches to this release are incorporated in the r-patched snapshot build.
- A build of the development version (which will eventually become the
- Previous releases

Note to webmasters: A stable link which will redirect to the current Windows
<CRAN MIRROR>/bin/windows/base/release.html.

Last change: 2021-02-15



R-4.0.4 for Windows (32/64 bit)

Download R 4.0.4 for Windows (85 megabytes, 32/64 bit)

Installation and other instructions
New features in this version

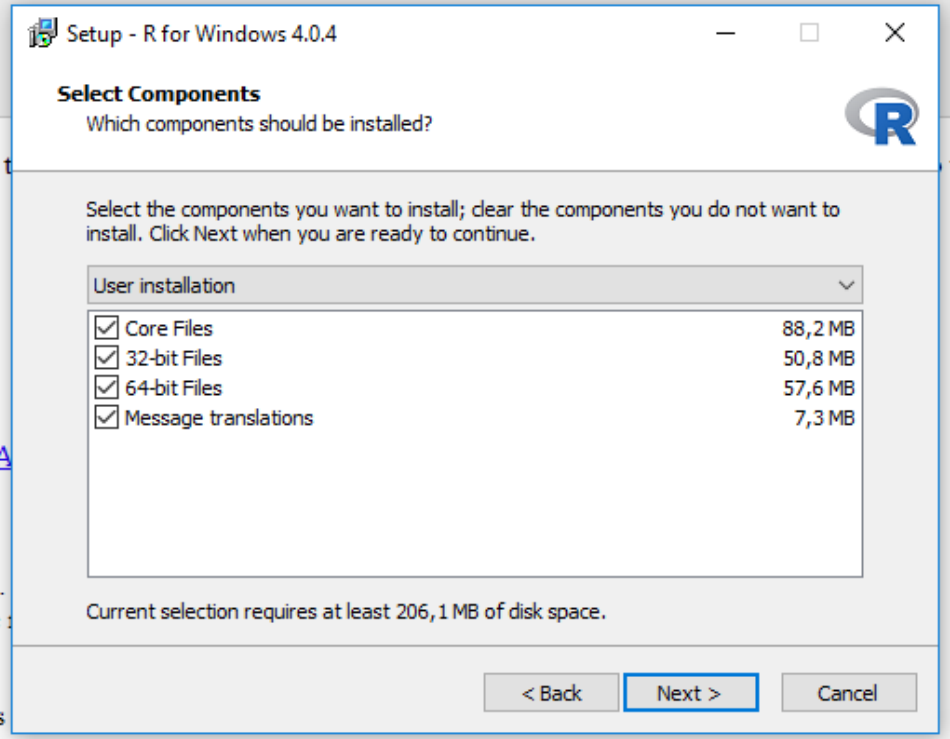
If you want to double-check that the package you have downloaded matches t the fingerprint or and command line versions are available.

- Does R run under my version of Windows?
- How do I update packages in my previous version of R?
- Should I run 32-bit or 64-bit R?

Please see the R FAQ for general information about R and the R Windows FA

- Patches to this release are incorporated in the r-patched snapshot build.
- A build of the development version (which will eventually become the
- Previous releases

Note to webmasters: A stable link which will redirect to the current Windows <CRAN MIRROR>/bin/windows/base/release.html.



Last change: 2021-02-15

R-4.0.4 for Windows (32/64 bit)

Download R 4.0.4 for Windows (85 megabytes, 32/64 bit)

Installation and other instructions
New features in this version

If you want to double-check that the package you have downloaded matches t the fingerpr
and command line versions are available.

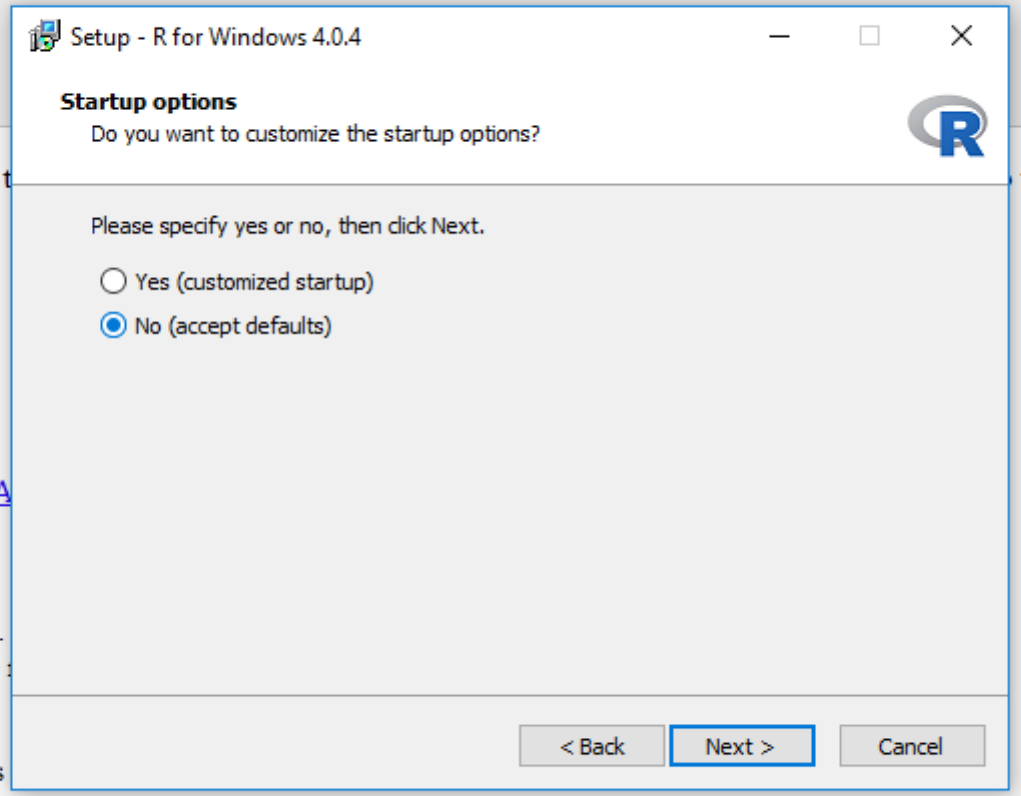
- Does R run under my version of Windows?
- How do I update packages in my previous version of R?
- Should I run 32-bit or 64-bit R?

Please see the R FAQ for general information about R and the R Windows FA

- Patches to this release are incorporated in the r-patched snapshot build.
- A build of the development version (which will eventually become the
- Previous releases

Note to webmasters: A stable link which will redirect to the current Windows
<CRAN MIRROR>/bin/windows/base/release.html.

Last change: 2021-02-15



R-4.0.4 for Windows (32/64 bit)

Download R 4.0.4 for Windows (85 megabytes, 32/64 bit)

Installation and other instructions
New features in this version

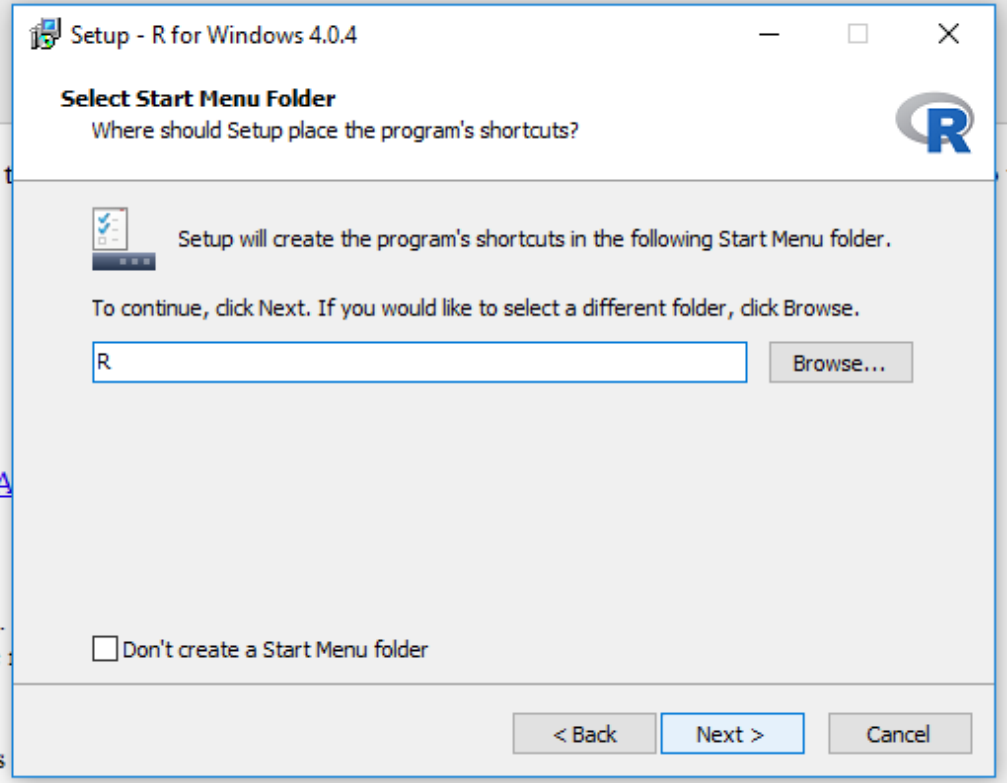
If you want to double-check that the package you have downloaded matches t the fingerpr
and command line versions are available.

- Does R run under my version of Windows?
- How do I update packages in my previous version of R?
- Should I run 32-bit or 64-bit R?

Please see the R FAQ for general information about R and the R Windows FA

- Patches to this release are incorporated in the r-patched snapshot build.
- A build of the development version (which will eventually become the
- Previous releases

Note to webmasters: A stable link which will redirect to the current Windows
<CRAN MIRROR>/bin/windows/base/release.html.



Last change: 2021-02-15

R-4.0.4 for Windows (32/64 bit)

Download R 4.0.4 for Windows (85 megabytes, 32/64 bit)

Installation and other instructions
New features in this version

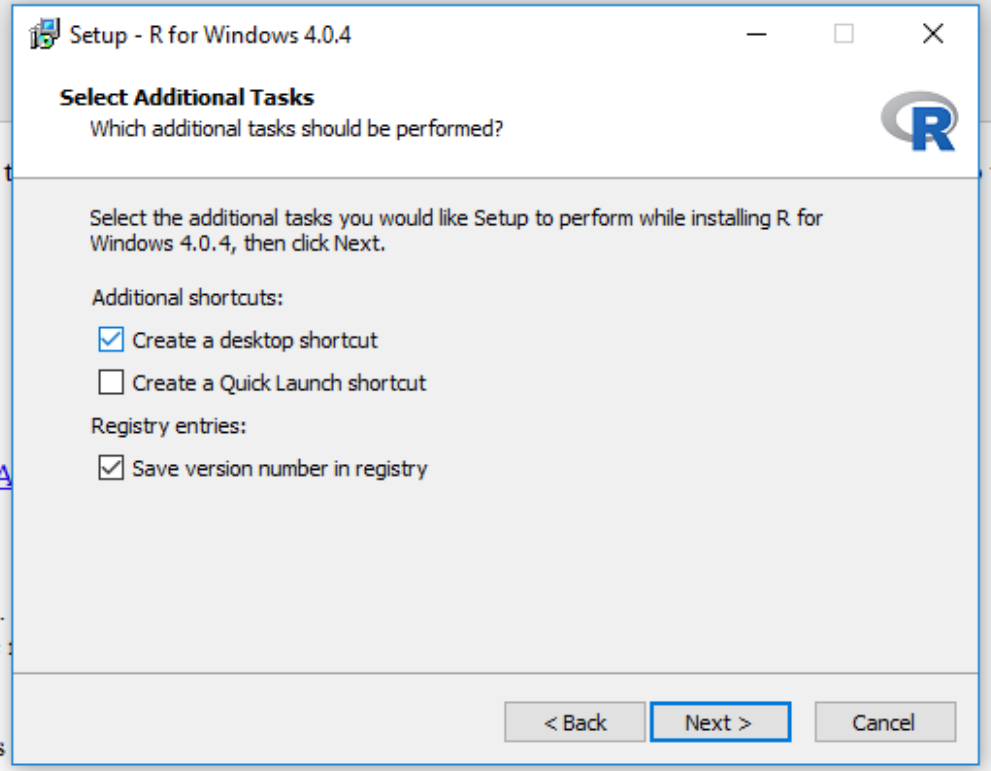
If you want to double-check that the package you have downloaded matches t ... the fingerpr
and command line versions are available.

- Does R run under my version of Windows?
- How do I update packages in my previous version of R?
- Should I run 32-bit or 64-bit R?

Please see the R FAQ for general information about R and the R Windows FA

- Patches to this release are incorporated in the r-patched snapshot build.
- A build of the development version (which will eventually become the
- Previous releases

Note to webmasters: A stable link which will redirect to the current Windows
<CRAN MIRROR>/bin/windows/base/release.html.



Last change: 2021-02-15

How R Works?

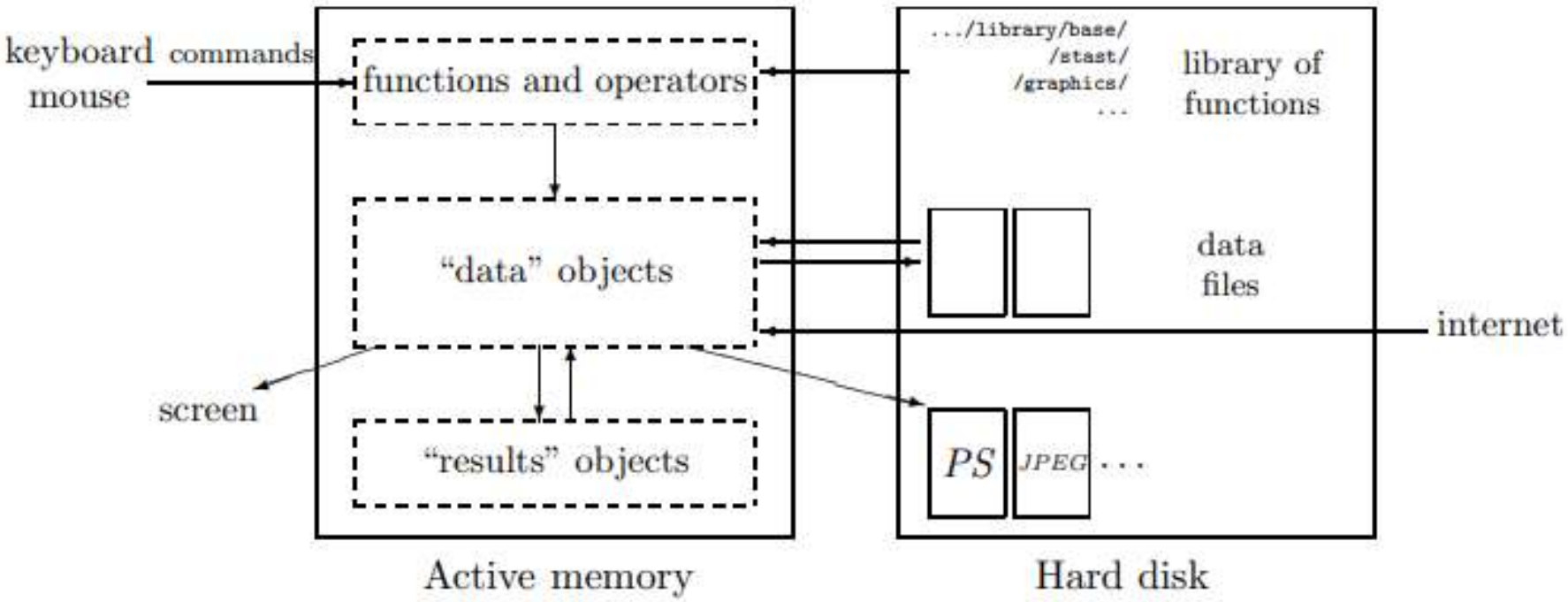


Figure 1: A schematic view of how R works.

Assignments

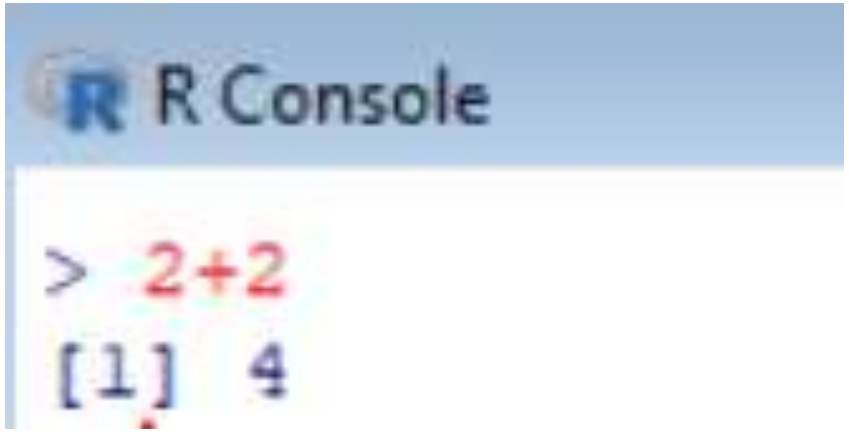
R, like other computer languages, has *symbolic variables*, that is names that can be used to present values. To assign the value 2 to the variable x,

```
>x<-2
```

The two characters <- should be read as a single symbol: an arrow pointing to the variable to which value is assigned. This is know as the ***assignment operator***.

An Overgrown Calculator

One of the simplest possible tasks in R is to enter an arithmetic expression and receive a result. (The second line is the answer from the machine.)



For instance; here is how to compute e^{-2} :

```
> exp(-2)
[1] 0.1353353
```

The (1) in front of the results is part of R's way to printing numbers and vectors.

The number in the brackets is the index of the first number on that line.

Operators					
Arithmetic		Comparison		Logical	
+	addition	<	lesser than	! x	logical NOT
-	subtraction	>	greater than	x & y	logical AND
*	multiplication	<=	lesser than or equal to	x && y	id.
/	division	>=	greater than or equal to	x | y	logical OR
^	power	==	equal	x || y	id.
%%	modulo	!=	different	xor(x, y)	exclusive OR
%/%	integer division				

Let’s consider the case of generating 15 random numbers from a normal distribution:

```
> rnorm(15)
 [1]  1.68095057 -0.35025810 -1.98253884 -1.18246504  0.21686112 -0.39960118
 [7]  0.28085088  0.28729635 -1.16729717 -1.02058836  0.98067259 -1.03320634
[13] -1.45899903  1.61512306  0.04717287
```

Here , for example the (7) indicates that 0.28085088 is the seventh element in the vector.

Here there is no immediately visible results, but from now on, x has the value 2 and can be used in subsequent arithmetic expressions.

```
> x
[1] 2
> x+x
[1] 4
```

PACKAGES IN R

The following table lists the standard packages which are distributed with a base installation of R. Some of them are loaded in memory when R starts; this can be displayed with the function search:

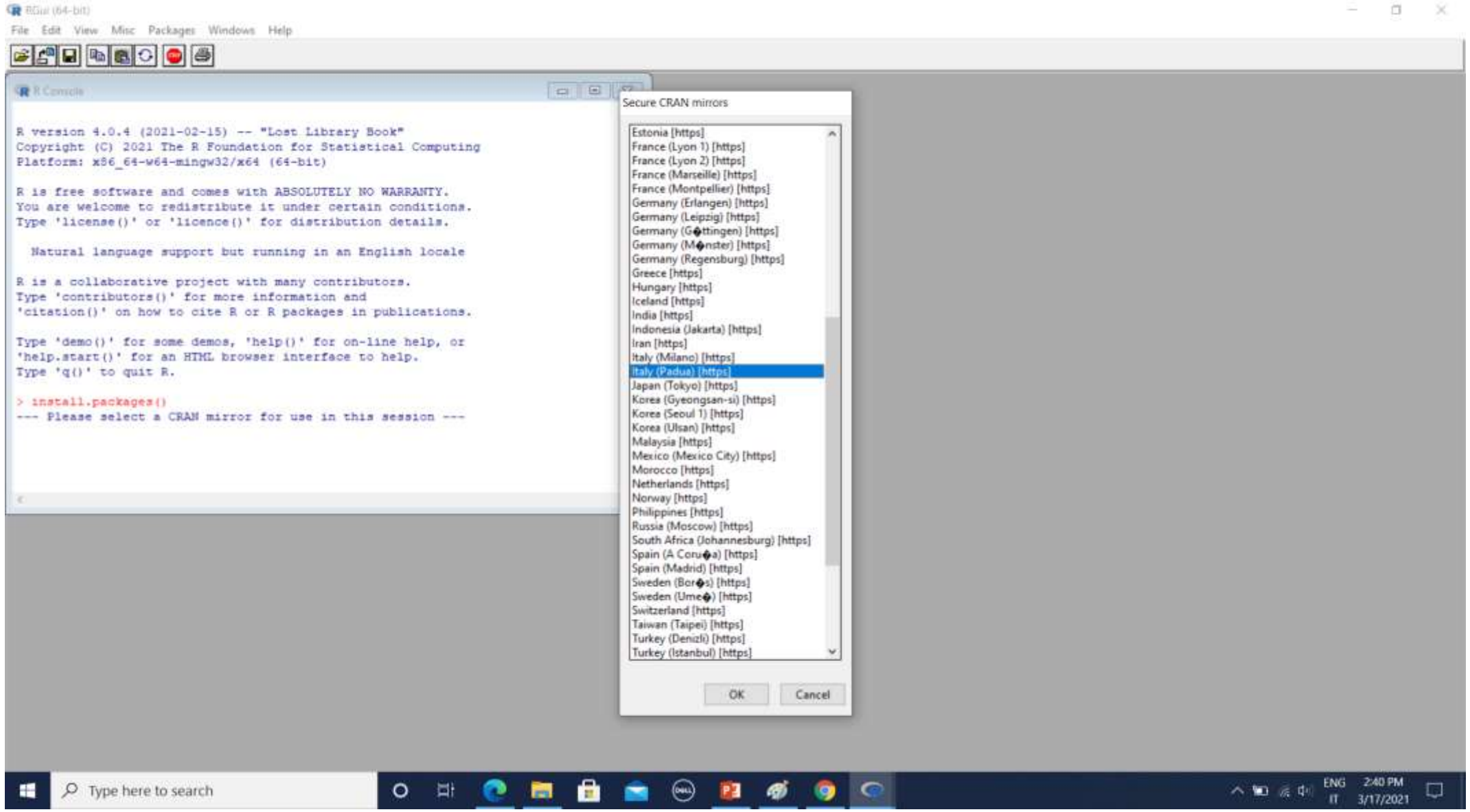
```
> search()
[1] ".GlobalEnv"        "package:stats"     "package:graphics"
[4] "package:grDevices" "package:utils"     "package:datasets"
[7] "package:methods"   "Autoloads"         "package:base"
```

Package	Description
base	base R functions
datasets	base R datasets
grDevices	graphics devices for base and grid graphics
graphics	base graphics
grid	grid graphics
methods	definition of methods and classes for R objects and programming tools
splines	regression spline functions and classes
stats	statistical functions
stats4	statistical functions using S4 classes
tcltk	functions to interface R with Tcl/Tk graphical user interface elements
tools	tools for package development and administration
utils	R utility functions

Package	Description
boot	resampling and bootstraping methods
class	classification methods
cluster	clustering methods
foreign	functions for reading data stored in various formats (S3, Stata, SAS, Minitab, SPSS, Epi Info)
KernSmooth	methods for kernel smoothing and density estimation (including bivariate kernels)
lattice	Lattice (Trellis) graphics
MASS	contains many functions, tools and data sets from the libraries of "Modern Applied Statistics with S" by Venables & Ripley
mgcv	generalized additive models
nlme	linear and non-linear mixed-effects models
nnet	neural networks and multinomial log-linear models
rpart	recursive partitioning
spatial	spatial analyses ("kriging", spatial covariance, ...)
survival	survival analyses

[18] http://cran.r-project.org/src/contrib/PACKAGES.html

How We Can Install Package?



RGui (64-bit)

File Edit View Misc Packages Windows Help

R Console

```
R version 4.0.4 (2021-02-15) -- "Lost Library Book"
Copyright (C) 2021 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> install.packages()
--- Please select a CRAN mirror for use in this session ---
```

Packages

IsoplotR
IsoplotRgui
ISOpureR
isoreader
IsoriX
IsoSpecR
isoSurv
isotone
isotonic.pen
IsotopeR
isotree
ISOweek
ispd
isqg
ISR3
ISRaD
istacr
iSTATS
isva
ISwR
italy
itan
itcSegment
itemanalysis
iteRates
iterators
iterLap
iterpc
itertools
itertools2
ITGM
ITNr
iTOP
itraxR
ITRLearn
ITRSelect
its.analysis
itsadug

OK Cancel

Type here to search

ENG IT 2:41 PM 3/17/2021

- help(package="ISwR")

Attach And Detach

The notation for accessing variables in data frames gets rather heavy if we repeatedly have to write longish commands like

Fortunately, we can make R look for objects among the variables in a given data frame, for example thuesen. We can write

```
> library(ISwR)
> attach(thuesen)
> intake
    pre post
1  5260 3910
2  5470 4220
3  5640 3885
4  6180 5160
5  6390 5645
6  6515 4680
7  6805 5265
8  7515 5975
9  7515 6790
10 8230 6900
```

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

VECTORS

The function vector, which has two arguments mode and length, creates a vector which elements have a value depending on the mode specified as argument: 0 if numeric, FALSE if logical, or "" if character.

The following functions have exactly the same effect and have for single argument the length of the vector: numeric(), logical(), and character().

A data vector is simply an array of numbers, and a vector variable can be constructed like this:

```
> weight <-c(60, 72, 57, 90, 95, 72)
> weight
[1] 60 72 57 90 95 72
```

The construct c(……) is used to define vectors.

Suppose that we also have the heights that correspond to the weights above. The body mass index (BMI) is defined for each person as the weight in kg divided by the square of the height in meters. This calculated as follows:

```
> height <- c(1.75, 1.80, 1.65, 1.90, 1.74, 1.91)
>
> bmi <- weight/height^2
> bmi
[1] 19.59184 22.22222 20.93664 24.93075 31.37799 19.73630
```

A Character vector is a vector of text strings, whose elements are specified and printed in quotes:

- >c<-(“Huey”, “Dewey”, “Louie”)
- (1) “Huey”, “Dewey”, “Louie”

Logical vectors can take a value TRUE or FALSE (or NA; see below).

Logical vectors are constructed using the c function just like the other vector types:

- >c(T,T,F,T)
- (1) TRUE TRUE FALSE TRUE
- (TALK HERE ABOUT MISSING VALUES)

Functions That Create Vectors

Here we introduce three functions ***c, seq,*** *and* ***rep,*** which are used to create vectors in various situations.

The first of these, c, has already been introduced.

```
> c (42, 57, 12, 39, 1, 3, 4)
[1] 42 57 12 39  1  3  4
```

The second function, ***seq*** **("sequence"),** is used for equidistant series of numbers.

```
> seq (4, 9)
[1] 4 5 6 7 8 9
```

```
> seq (4, 10, 2)
[1]  4  6  8 10
```

```
> 4:9
[1] 4 5 6 7 8 9
```

The third function, ***rep*** **("replicate"),** is used to generate repeated values. It is used in two variants, depending on whether the second argument is a vector or a single number:

```
> oops<-c(7, 9, 13)
> rep(oops, 3)
[1]  7  9 13  7  9 13  7  9 13
> rep(oops, 1:3)
[1]  7  9  9 13 13 13
```

Matrices

A matrix is actually a vector with an additional attribute (dim) which is itself a numeric vector with length 2, and defines the numbers of rows and columns of the matrix.

A matrix can be created with the function matrix:

matrix(data = NA, nrow = 1, ncol = 1, byrow = FALSE, dimnames = NULL)

Matrices are used for many purposes in theorical and practical statistics.

Matrices and also higher-dimensional arrays do get used for simpler purposes as well, mainly to hold tables, so an elementary description is in order. Matrices an arrays are represented as vectors with dimensions:

```
> x <- 1:12
> dim(x) <- c(3,4) (row, column)
Error: attempt to apply non-function
> x
 [1]  1  2  3  4  5  6  7  8  9 10 11 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
```

Notice how the byrow=T switch causes the matrix to be filled in a rowwise fashion rather than columnwise.

Useful functions that operate on matrices include ***rownames, colnames*** and the transposition function ***t***, which turns rows into columns, and vice versa.

```
> x <-matrix(1:12, nrow=3,byrow=T)
> rownames(x) <-LETTERS[1:3]
> x
  [,1] [,2] [,3] [,4]
A    1    2    3    4
B    5    6    7    8
C    9   10   11   12
```

```
> t(x)
     A B  C
[1,] 1 5  9
[2,] 2 6 10
[3,] 3 7 11
[4,] 4 8 12
```

We can see “glue” vectors together, columnwise or rowwise, using the ***cbind*** and ***rbind*** functions.

```
> cbind(A=1:4, B=5:8, C=9:12)
     A B  C
[1,] 1 5  9
[2,] 2 6 10
[3,] 3 7 11
[4,] 4 8 12
>
>
>
> rbind(A=1:4, B=5:8, C=9:12)
  [,1] [,2] [,3] [,4]
A    1    2    3    4
B    5    6    7    8
C    9   10   11   12
```

For matrices and data frames, ***colnames*** and ***rownames*** are labels of the columns and rows, respectively. They can be accessed either with their respective functions, or with ***dimnames*** which returns a list with both vectors.

```
> X <- matrix(1:4, 2)
> rownames(X) <- c("a", "b")
> colnames(X) <- c("c", "d")
> X
  c d
a 1 3
b 2 4
```

FACTOR

A factor includes not only the values of the corresponding categorical variable, but also the different possible levels of that variable (even if they are not present in the data). The function factor creates a factor with the following options:

>**factor(x, levels = sort(unique(x), na.last = TRUE), labels = levels, exclude = NA, ordered = is.ordered(x))**

levels specifies the possible levels of the factor (***by default the unique values of the vector x***), labels defines the names of the levels, exclude the values of x to exclude from the levels, and ordered is a logical argument specifying whether the levels of the factor are ordered.

Recall that x is of mode numeric or character.

```
> factor(1:3)
[1] 1 2 3
Levels: 1 2 3
> factor(1:3, labels=c("A", "B", "C"))
[1] A B C
Levels: A B C
```

The terminology is that a factor has a set levels-say four levels for concreteness.

```
> pain<-c(0,3,2,2,1)
> fpain<-factor(pain, levels=0:3)
> levels(fpain)<-c("none" , "mild" , "medium" , "severe")
> fpain
[1] none   severe medium medium mild
Levels: none mild medium severe
```

The first command creates a numerical vector pain, encoding the pain level of five patients.

The effect of the final line is that the level names are changed to the four specified character strings. The result should be apparent from the following:

```
> as.numeric(fpain)
[1] 1 4 3 3 2
> levels(fpain)
[1] "none"   "mild"   "medium" "severe"
```

DATA FRAME

A data frame corresponds to what other statistical packages call a “datamatrix” or a “data set”.

It is a list of vectors and/or factors of the same length, which are related “across”, such data in the same position come from the same experimental unit (subject, animal, etc.). In addition , it has a unique set of row names.

To create data frames from preexisting variables:

```
> intake.pre<-c(5260, 5470,5640, 6180, 6390, 6515, 6805, 7515, 7515, 8230, 8770)
> intake.post<-c(3910, 42240, 3885, 5160, 5645, 4680, 5265, 5975, 6790, 6900, 7335)
> d<-data.frame(intake.pre, intake.post)
> d
   intake.pre intake.post
1        5260        3910
2        5470       42240
3        5640        3885
4        6180        5160
5        6390        5645
6        6515        4680
7        6805        5265
8        7515        5975
9        7515        6790
10       8230        6900
11       8770        7335
```

As with lists, variables are accessible using the **$** notation:

```
> d$intake.pre
 [1] 5260 5470 5640 6180 6390 6515 6805 7515 7515 8230 8770
> intake.pre
 [1] 5260 5470 5640 6180 6390 6515 6805 7515 7515 8230 8770
```

Indexing

If we need a particular element in a vector, for instance the premenstrual energy intake for woman no.5, we can do

```
> intake.pre[5]
[1] 6390
```

It is also worth noting that to get a sequence of elements, for instance, the first five, we can use the a:b notation:

```
> intake.pre[1:5]
[1] 5260 5470 5640 6180 6390
```

Conditional Selection

```
> intake.post[intake.pre>7000]
[1] 5975 6790 6900 7335
> intake.post[intake.pre>7000 & intake.pre<=8000]
[1] 5975 6790
```

Indexing Of Data Frames

It is also possible to use a notation that uses the matrix–like structure directly:

```
> d
   intake.pre intake.post
1        5260        3910
2        5470       42240
3        5640        3885
4        6180        5160
5        6390        5645
6        6515        4680
7        6805        5265
8        7515        5975
9        7515        6790
10       8230        6900
11       8770        7335
> d[5,1]
[1] 6390
```

```
> d[d$intake.pre>7000]
Error in `[.data.frame`(d, d$intake.pre > 7000) :
  undefined columns selected
> d[d$intake.pre>7000, ]
   intake.pre intake.post
8        7515        5975
9        7515        6790
10       8230        6900
11       8770        7335
```

Subset

A couple of functions exist to make things a little easier. They are used as follows(data is used to fetch a built-in data set):

```
> data(thuesen)
> thue2 <-subset(thuesen, blood.glucose<7)
> thue2
   blood.glucose short.velocity
6            5.3           1.49
11           6.7           1.25
12           5.2           1.19
15           6.7           1.52
17           4.2           1.12
22           4.9           1.03
```

Grouped Data And Data Frames

The natural way of storing grouped data in a data frame is to have the data themselves in one vector and parallel to that to have a factor telling which data are from which group.

```
> data(energy)
> energy
   expend stature
1    9.21   obese
2    7.53    lean
3    7.48    lean
4    8.08    lean
5    8.09    lean
6   10.15    lean
7    8.40    lean
8   10.88    lean
9    6.13    lean
10   7.90    lean
11  11.51   obese
12  12.79   obese
13   7.05    lean
14  11.85   obese
15   9.97   obese
16   7.48    lean
17   8.79   obese
18   9.69   obese
19   9.68   obese
20   7.58    lean
21   9.19   obese
```

```
> exp.lean <-energy$expend[energy$stature=="lean"]
> exp.obese <-energy$expend[energy$stature=="obese"]
> l <-split(energy$expend, energy$stature)
> l
$lean
 [1]  7.53  7.48  8.08  8.09 10.15  8.40 10.88  6.13  7.90  7.05  7.48  7.58  8.11

$obese
[1]  9.21 11.51 12.79 11.85  9.97  8.79  9.69  9.68  9.19
```

PROBABILITY AND DISTRIBUTIONS

The concepts of randomness and probability are central to statistics. It is an empirical fact that most experiments and investigations are not perfectly reproducible.

In this section we outline the basic ideas of probability and the functions that R has for random sampling and handling of theoretical distributions.

Population, the totality of all subjects possessing certain common characteristics that are being studied.

Sample; a subgroup or subset of the population.

A sample obtained without bias or showing preferences in selecting items of the population is called a ***random sample***.

Random Sampling

In R we can simulate these situations with the ***sample*** function.

If we want to pick five numbers at random from the set 1:40, then we can write

```
> sample (1 : 40, 5)
[1] 37 24 39 15 29
```

What is a Distribution?
Reminder of Basic Definitions

A discrete random variable is a numerical quantity that takes values with some randomness from a discrete set; often a subset of integers.

The probability distribution of a discrete random variable specifies the probability associated with each possible value.

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)`

Four fundamental items can be calculated for a statistical distribution:

- ✓Density or point probability
- ✓Cumulated probability, distribution function
- ✓Quantiles
- ✓Pseudo-random numbers

For all distributions implemented in R, there is a function for each of the four items listed above.

For example, for the normal distribution, these are named **dnorm, pnorm, qnorm, and rnorm,** respectively (density, probability, quantile, and random).

What is a Distribution?
Continuous Random Variables

A continuous random variable X takes values in an interval of real numbers. There is a probability associated with X falling between two numbers $a < b$. The density function $f_X(x)$ is such that $Prob(a \leq X \leq b)$ is the area bounded by the graph of $y = f_X(x)$, the $x-$axis, and the vertical lines $x = a$ and $x = b$. In other words, $Prob(a \leq X \leq b) = \int_a^b f_X(x)dx$.

Normal Distribution

A normally distributed random variable with mean μ and standard deviation σ is one with density function fX (x) = 1 √ 2πσ e − (x−μ) 2 2σ2 . It's graph is a bell curve centered at μ whose "spread" is determined by σ.

The standard normal distribution is one with mean 0 and standard deviation 1.

Normal Distribution Computing Values in R

The distribution function for the normal with mean = ’mean’ and standard deviation = ’sd’ is **pnorm(x, mean, sd).**

The quantile function of the normal is qnorm(p, mean, sd).

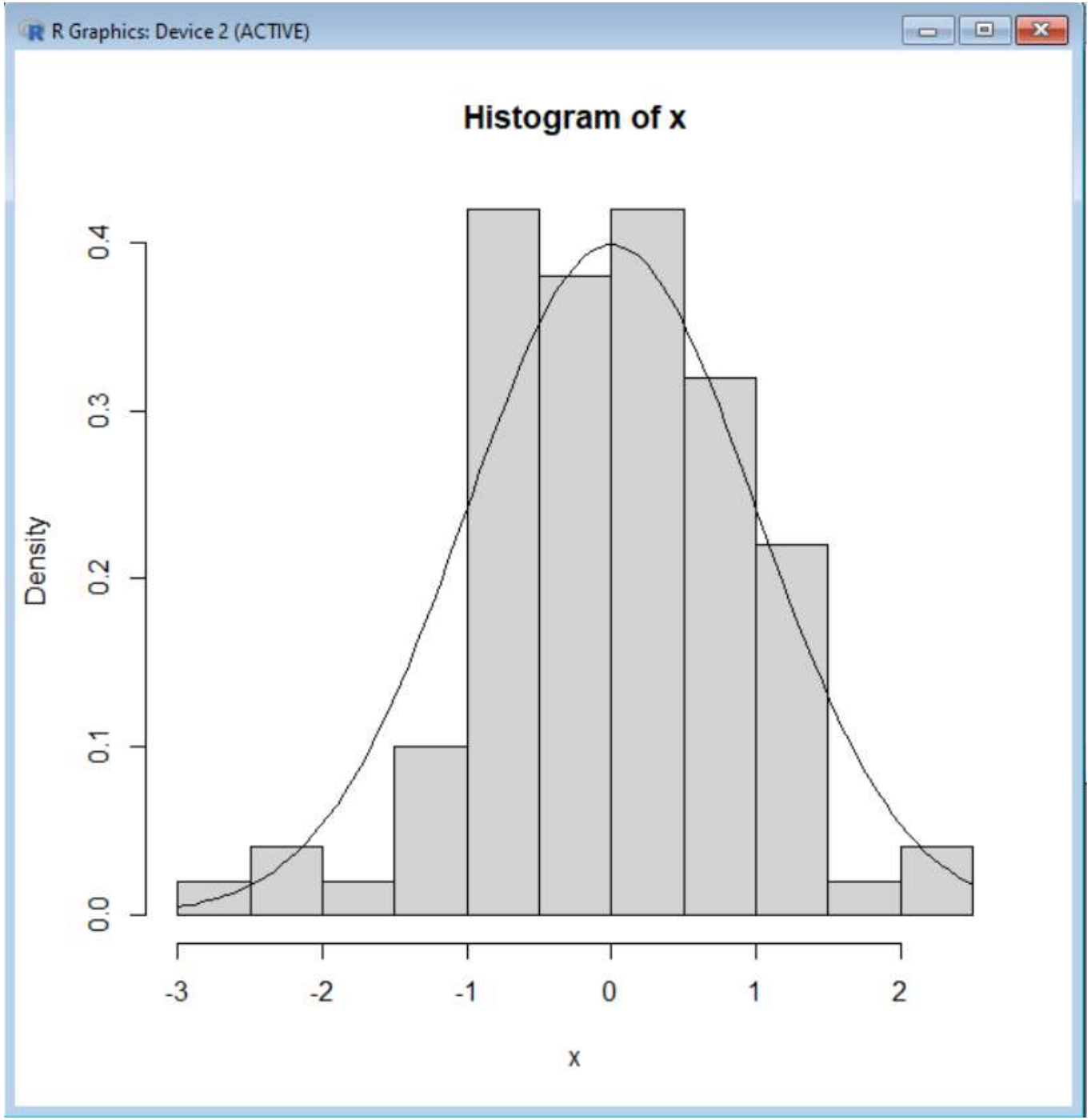
The function **rnorm(n, mean, sd)** randomly generates n values of a normally distributed random variable with given mean and sd.

Default values: mean=0, sd=1.

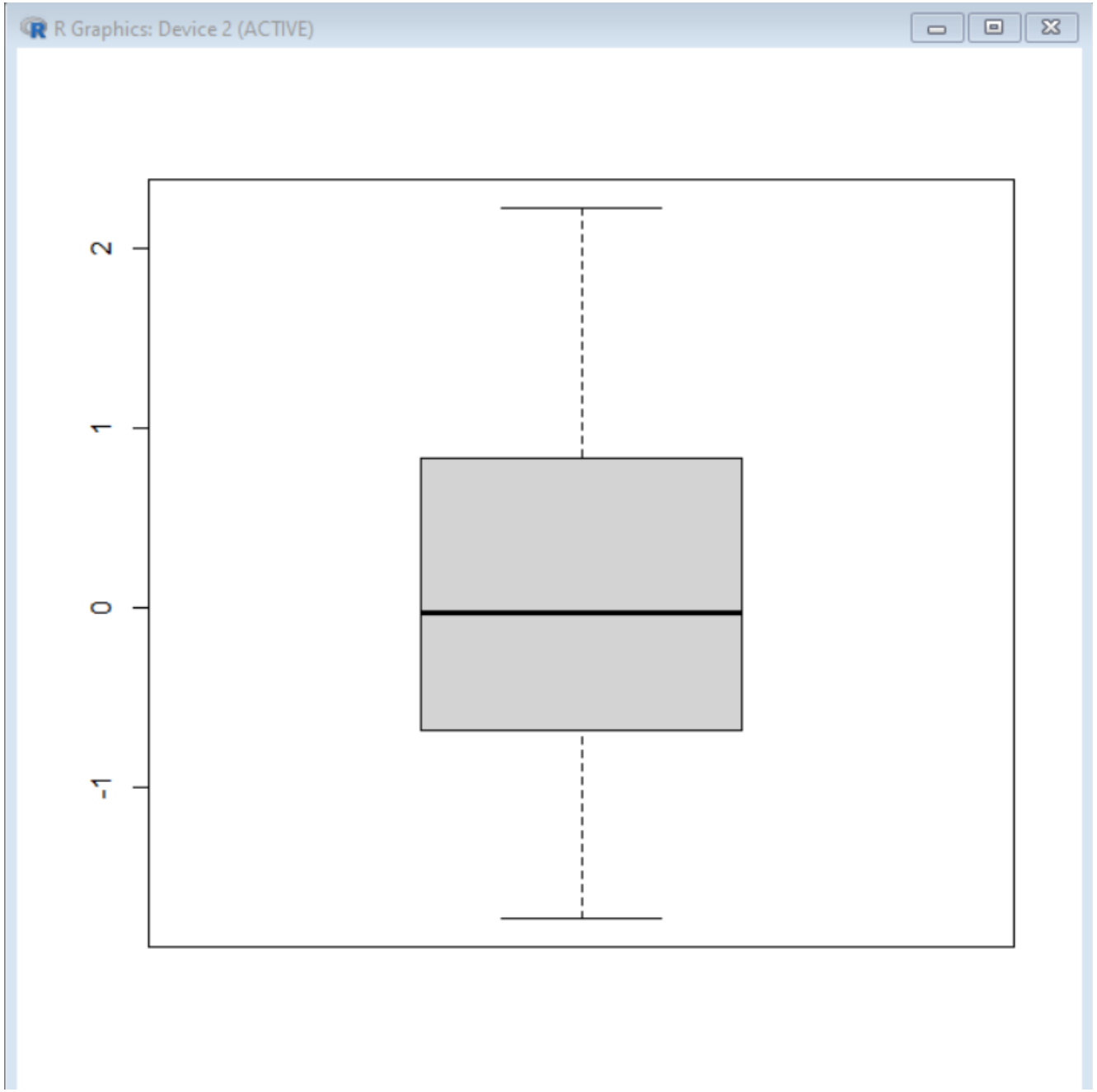
Normal Distribution

```
> pnorm(2)
[1] 0.9772499
> pnorm(0)
[1] 0.5
> qnorm(0.95)
[1] 1.644854
> qnorm(0.025, mean = 2, sd = 0.5)
[1] 1.020018
> rnorm(4, 2, 2)
[1]  3.619208  6.409759  1.790710 -1.171351
>
```

```
> x<-rnorm(100)
> hist(x, freq=F)
> curve(dnorm(x), add=T)
```



```
> x<-rnorm(100)
> boxplot(x)
```



```
> x<-0:50
> plot(x, dbinom(x, size=50, prob=.33), type="h")
```