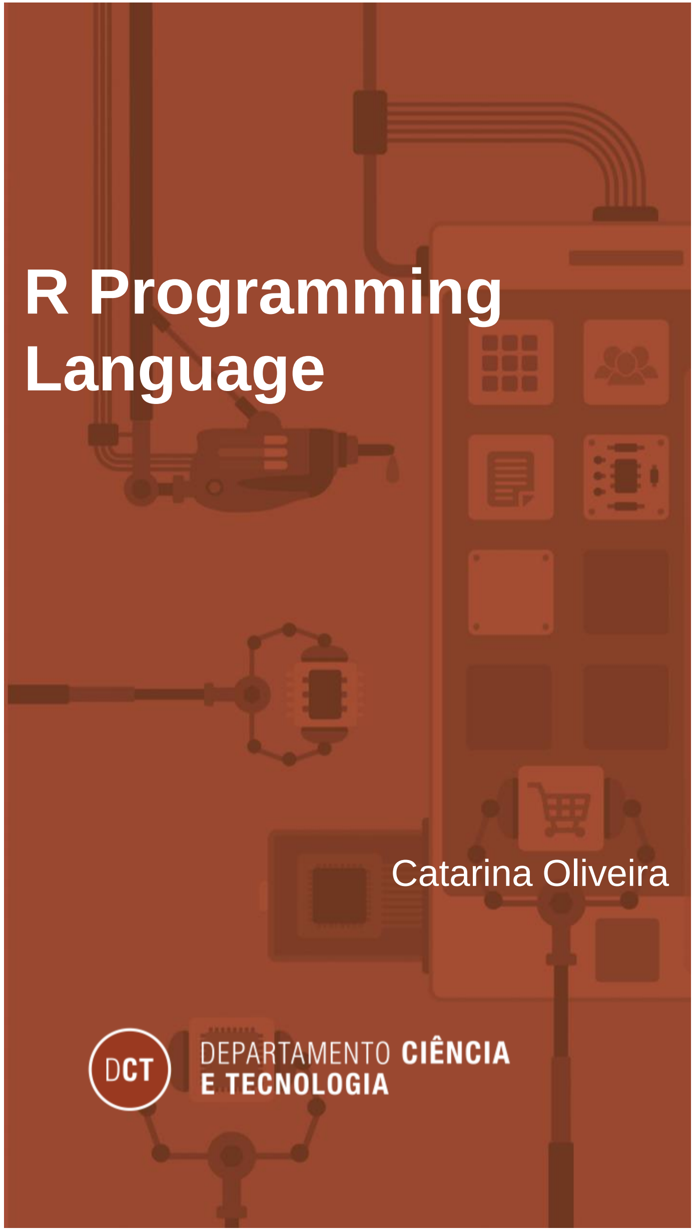


R Programming Language

Catarina Oliveira

DEPARTAMENTO CIÊNCIA
E TECNOLOGIA

UNIVERSIDADE PORTUCALENSE

CONTENTS

1. R
2. Environment installation
3. Development environment
4. Useful commands
5. Variables
6. constants
7. operators
    1. Assignment
    2. arithmetic
    3. Relational
    4. logical
8. precedence and associativity
9. complex objects
    1. Vector
    2. Headquarters
    3. List
    4. data frame
    5. Factor
10. Predefined Functions in R
    1. Numerical
    2. Text
    3. Statistics
    4. Probability
    5. Useful

R

R is a language and environment for statistical computing and graphing

Includes:

- Ability to handle and store data effectively
- A set of operators for performing vector and matrix calculations
- A large, integrated set of data analysis tools
- Graphical capabilities for data analysis and visualization
- Conditional, repetitive structures
- Possibility to define functions
- Input and output capability

Environment installation

Windows

- A: https://cran.r-project.org/bin/windows/
- Rstudio : https://download1.rstudio.org/desktop/windows/RStudio-1.3.1093.exe

Mac

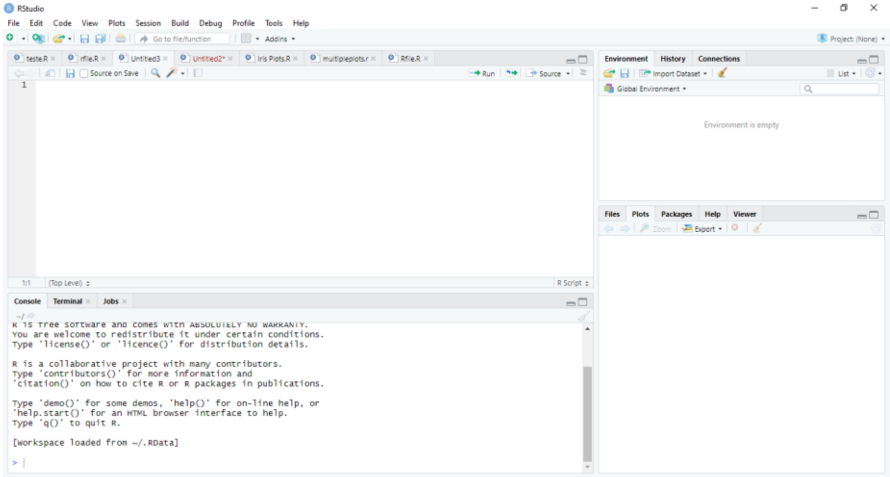
- A: https://cran.r-project.org/bin/macosx/
- Rstudio : https://download1.rstudio.org/desktop/macos/RStudio-1.3.1093.dmg

Linux

- A: https://cran.r-project.org/bin/linux/
- Rstudio (depends on Linux version)

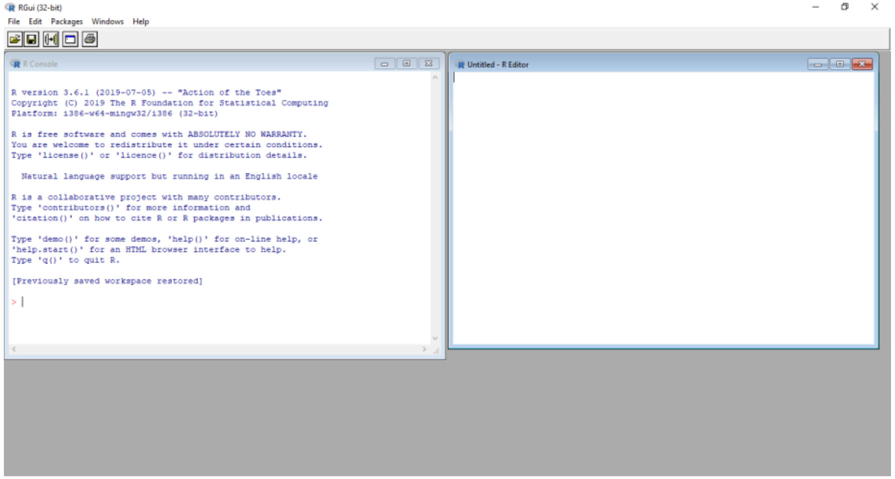
Development environment

RStudio



```
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.

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.

[Workspace loaded from ~/.RData]

> |
```

R GUI



```
R version 3.6.1 (2019-07-05) -- "Action of the Toes"
Copyright (C) 2019 The R Foundation for Statistical Computing
Platform: i386-w64-mingw32/i386 (32-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.

[Previously saved workspace restored]

> |
```

Useful commands

workspace

```
getwd () # shows the folder we are working
setwd ("path") # defines the folder we are working
```

Variables

```
ls() # shows the list of objects ( variables , functions )
rm( object_name ) # remove the object from the environment called object_name
```

Examples and help

```
example ( package_name ) # show package examples _ called package_name
help ( package_name ) # show package help _ called package_name
? function_name # show help for function call function_name
```

cheat Basic sheet : https://rstudio.com/wp-content/uploads/2016/10/r-cheat-sheet-3.pdf

cheat Sheets : https://rstudio.com/resources/cheatsheets/

Variables

Variable: memory location used to store information. Information can be changed

Examples

- Valid variable names

```
division
Square
. sub.multiplication # start with dot followed by letter . Variable invisible to ls()
accumulative_sum
Sum5
```

- Invalid variable names

```
tot@ l # character usage specials
5um # start with a number
_fine # start with underscore
FALSE # word reserved
.0three # start with a dot followed by number
```

constants

Constant: memory location used to store information. Information can **not** be changed

Examples

- Numerical

```
typeof (2)
typeof (2L) # L - long ( integers large )
typeof (2i) # numbers complexes
```

- Characters (independent of using single quotes or quotation marks)

```
typeof ('example')
typeof ("2")
```

operators

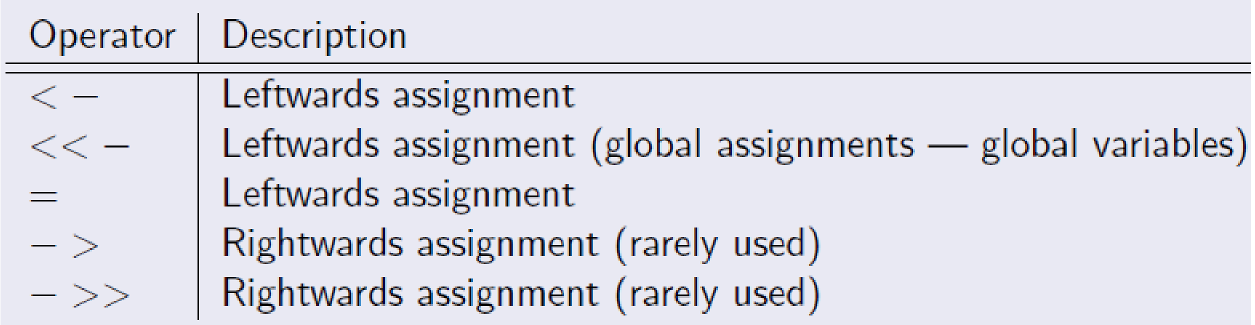
OR has several operators to perform different operations

Types:

- Assignment
- arithmetic
- Relational
- logical

assignment operators

Used to assign values to variables



Operator	Description
< −	Leftwards assignment
<< −	Leftwards assignment (global assignments — global variables)
=	Leftwards assignment
− >	Rightwards assignment (rarely used)
− >>	Rightwards assignment (rarely used)

Global variables : declared outside any function and can be accessed in any program function

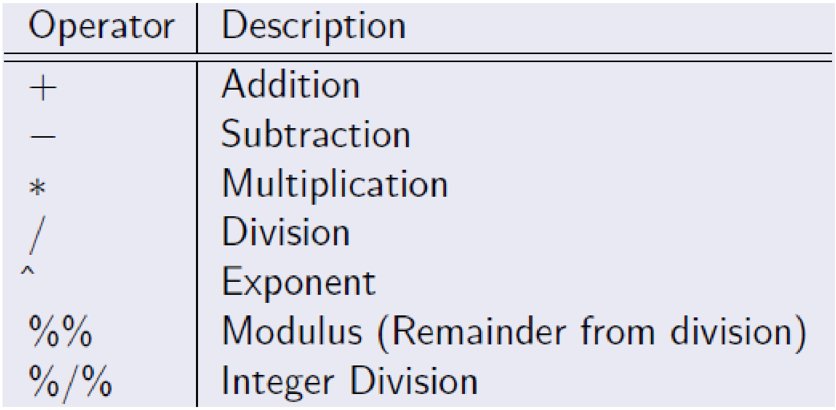
Local variables : declared within a function, and can only be used within that function

Examples

```
x <- 20

x = 30

5 -> x
```

arithmetic operators

Used for arithmetic operations



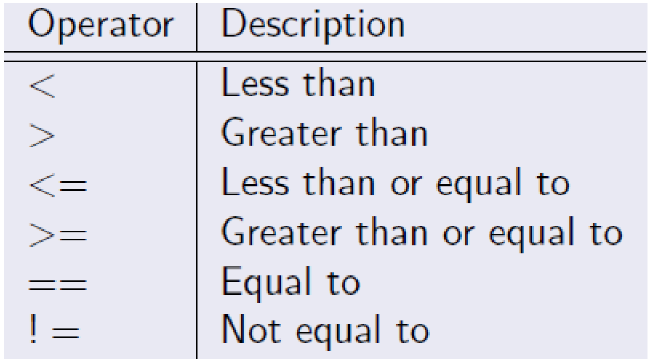
Operator	Description
+	Addition
−	Subtraction
*	Multiplication
/	Division
^	Exponent
%%	Modulus (Remainder from division)
%/%	Integer Division

Examples

```
x <- 20
y <- 15
x + y
x - y
x * y
y / x
y %/% x
y %% x
y^x
```

relational operators

They are used to make comparisons between values



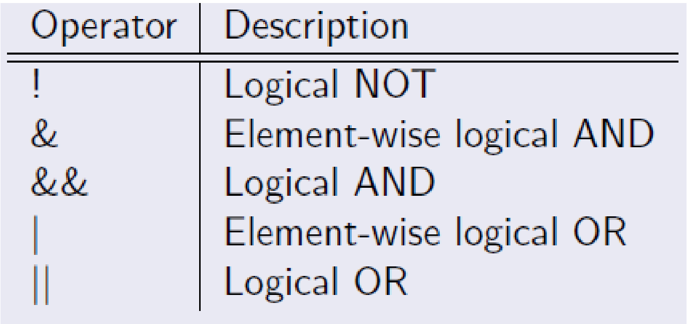
Operator	Description
<	Less than
>	Greater than
<=	Less than or equal to
>=	Greater than or equal to
==	Equal to
! =	Not equal to

Examples

```
x <- 20
y <- 15
x < y
x > y
x <= 15
```

Logical Operators

Used to perform logical operations



Operator	Description
!	Logical NOT
&	Element-wise logical AND
&&	Logical AND
|	Element-wise logical OR
||	Logical OR

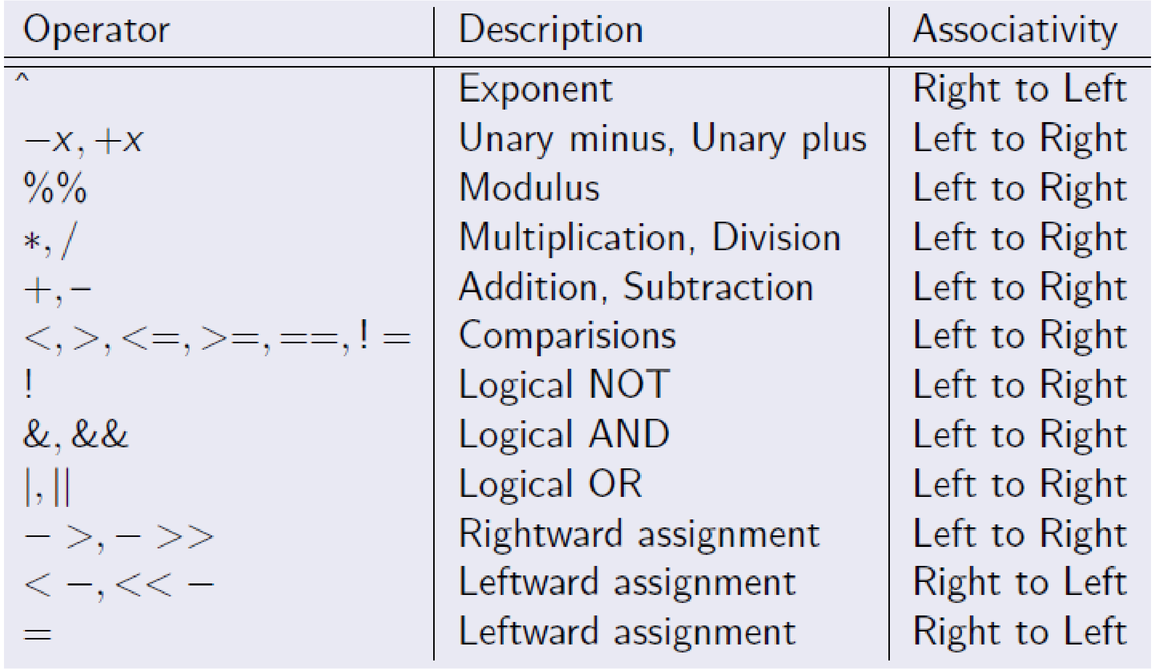
Logical operator " element-wise " : combines each element of the first vector with the corresponding element of the second vector and returns the output

Logical operator : uses the first element of each vector

Examples

```
x <- c(TRUE , FALSE ,0 ,3)
y <- c(FALSE ,TRUE ,FALSE , TRUE )
!x
x & y
x || and
```

precedence and associativity



Operator	Description	Associativity
^	Exponent	Right to Left
−x, +x	Unary minus, Unary plus	Left to Right
%%	Modulus	Left to Right
∗, /	Multiplication, Division	Left to Right
+, −	Addition, Subtraction	Left to Right
<, >, <=, >=, ==, ! =	Comparisions	Left to Right
!	Logical NOT	Left to Right
&, &&	Logical AND	Left to Right
|, ||	Logical OR	Left to Right
− >, − >>	Rightward assignment	Left to Right
< −, << −	Leftward assignment	Right to Left
=	Leftward assignment	Right to Left

Examples

precedence Associativity

```
3 + 4 / 2              3 / 4 / 2
(3 + 4) / 2            3 / (4 / 2)
```

complex objects

OR relies on different types of complex objects

Categories

- Vector
- Headquarters
- List
- *data frame*
- *Factor*

Vector

An array is a basic type of data structure that contains elements of the same type. Data types can be logical, integer, real, characters, etc.

Vectors are created using the c() function

The size of an array can be obtained using the length () function

The type of an array can be obtained using the typeof () function

Examples with vectors

Creating vectors using the c() function

```
x <- c(1, 5, 4, 9, 0)
x <- c(1, 5.4 , TRUE , " hello ")
```

Creating vectors using the “:” operator

```
x <- 1:7
y <- 2:-2
```

Creating vectors using the seq () function

```
seq (1, 3, by =0.2) # by range
seq (1, 5, length.out =4) # at output
```

Functions for vectors

```
length(x) # shows the number of elements in the vector
sort(x, decreasing = TRUE) # in order decreasing . For ascending use decreasing = TRUE
unique(x) # show values vector
table(x) # calculate vector
```

Examples with vectors

Reading using logical vector as index

```
x <- seq (-3, 3, 2)
x[c(TRUE , FALSE , FALSE , TRUE )]
x[x < 0]
x[x > 0]
```

Reading using character vector as index

```
x <- c(first=3, second=0, third=9)
names (x)
x["second"]
x[c(" first ", " third ")]
```

Examples with vectors

modify a vector

```
x <- seq (-3, 2, 1)
x[2] <- 0
x[x <0] <- 5
x <- x [1:4]
```

delete a vector

```
rm(x) # remove variable
```

Headquarters

An array is a two-dimensional data structure

The array is similar to the vector but still contains the dimension attribute which can be checked with the attributes () function

Examples with arrays

Create an array using the matrix () function

```
a <- matrix (1:9 , nrow = 3, ncol = 3)
b <- matrix (1:9 , nrow = 3)
c <- matrix (1:9 , nrow =3, byrow = TRUE )

# Change column and row names
x <- matrix (1:9 , nrow = 3, dimnames = list (c("X","Y","Z"), c("A","B","C")))
columns (x)
rownames (x)
colnames (x) <- c("C1","C2","C3")
rownames (x) <- c("R1","R2","R3")
```

Array creation using cbind () and rbind () functions

```
a <- cbind (c(1 ,2 ,3) ,c(4 ,5 ,6))
b <- rbind (c(1 ,2 ,3) ,c(4 ,5 ,6))
```

Array creation using dim () function

```
x <- c(1,2,3,4,5,6)
dim(x) <- c(2 ,3)
```

Examples with arrays

Reading using an array of integers as index

```
x <- matrix (1:9 , nrow = 3, ncol = 3)

x[1,] # select first row
x[, 1] # select first column
x[,] # leaving row as well as column field blank will select entire matrix
x[-1 ,] # select all rows except first

x[c(1 ,2) ,c(2 ,3)] # select rows 1 & 2 and columns 2 & 3
x[c(1 ,2) ,] # leaving column field blank will select entire columns
```

Examples with arrays

Reading using an array of logical values as an index

```
x[c(TRUE ,FALSE , TRUE ),c(TRUE ,TRUE , FALSE )]
x[c(TRUE , FALSE, TRUE ),c(2 ,3)]
x[x >5] # select elements greater than 5
x[x%%2 == 0] # select even elements
```

Reading using an array of characters as index

```
colnames (x) <- c("A","B","C")
shah"]
x[,c("A","C")]
x[2:3 ,c("A","C")]
```

Examples with arrays

modify an array

```
x[2,2] <-10; # modify a single element
x[x <5] <- 0; # modify elements less than 5
t(x) # transpose the matrix
cbind (x, c(1, 2, 3)) # add column
rbind (x, c(1 ,2 ,3)) # add row
x<-x[1:2,]; # remove the third row
```

delete an array

```
rm(x) # remove variable
```

List

Data structure with elements of various data types

Examples:

Create a list using the list () function

```
x <- list ("a" = 2.5 , "b" = TRUE , "c" = 1:3)
str(x) # check list structure
x <- list (2.5 , TRUE ,1:3) # tags are optional
```

List

Read elements from a list

```
x[c (1:2) ] # index using integer vector
x[ -2] # using negative integer to exclude second component
x[c(T,F,F)] # index using logical vector
x <- list (" name " = " John ", " age" = 19 , " speaks " = c(“ English "," French "))
x[c(" age"," speaks ")] # index using character vector
x["age"]
typeof (x["age"]) # single [ returns a list
x[["age"]] # double [[ returns the content
typeof (x[["age" ]])
x$name # same as x[[" name "]]
x$a # partial matching , same as x$ag or x $age
x[["a"]] # cannot do partial match with [[

# indexing can be done recursively
x$ speaks [1]
x[[" speaks " ]][2]
```

List

modify a list

```
x[[" name "]] <- "Clair"
x[[" married "]] <- FALSE
```

delete a list

```
rm (x)
```

data frame

A *data frame* is a two-dimensional data structure.

It's a special case of a list with all components the same size

Each component forms a column, and the contents of the components form the rows.

Examples:

Create a data frame using the data.frame () function

```
x <- date . frame(SN = 1:2 , Age = c (21 ,15) , Name = c(“John "," Dora "))

str(x) # structure of x

x <- date . frame("SN" = 1:2 , " Age" = c(21 ,15) , " Name " = c(" John ", " Dora "),
 stringsAsFactors = FALSE )

str(x) # now the third column is a character vector
```

data frame

read data frames

```
x["Name"]
x$ Name
x[["Name"]]
x [[3]]
str( trees ) # access as a matrix
head (trees ,n =3) # access as a matrix
trees [2:3 ,] # select 2nd and 3rd row
trees [ trees $ Height > 82 ,] # selects rows with Height greater than 82
trees [10:12 ,2]
trees [10:12 ,2 , drop = FALSE ]
```

data frame

Modify a data frame

```
x[1,"Age"] <- 20
cbind (x, State =c("NY","FL"))
```

Delete a data frame

```
rm (x)
```

factors

A *factor* is a structure used for fields that can only take a finite number of values (categorical)

Examples:

Create a factor using the factor () function

```
x <- factor (c(" single ", " married ", " married ", " single "));

x <- factor (c(" single ", " married ", " married ", " single "), levels = c(" single ", "
married", "divorced"));

x <- factor (c(" single "," married "," married "," single "))

str(x)
```

Factor

read a factor

```
x[3] # access 3rd element
x[c(2, 4)] # access 2nd and 4th element
x[ -1] # access all but 1st element
x[c(TRUE , FALSE , FALSE , TRUE )] # using logical vector
```

modify a factor

```
x[2] <- "single"
x [2] <- " divorced " # modify second element ; x
x [3] <- " widowed " # cannot assign values outside levels
```

delete a factor

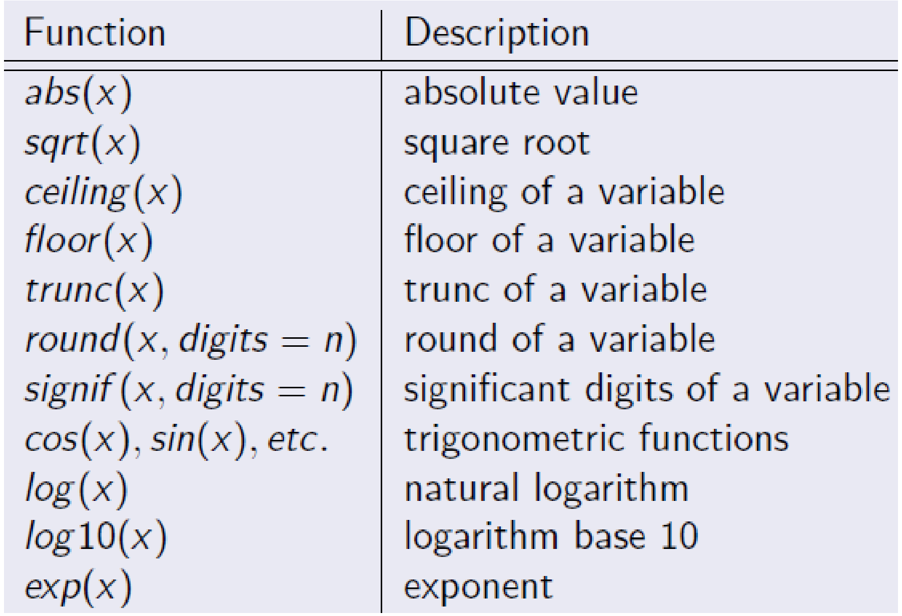
```
rm (x)
```

R predefined functions

OR has several predefined functions, which can be classified into the following categories:

- Numerical functions
- text functions
- statistical functions
- probability functions
- Useful functions

Numerical functions

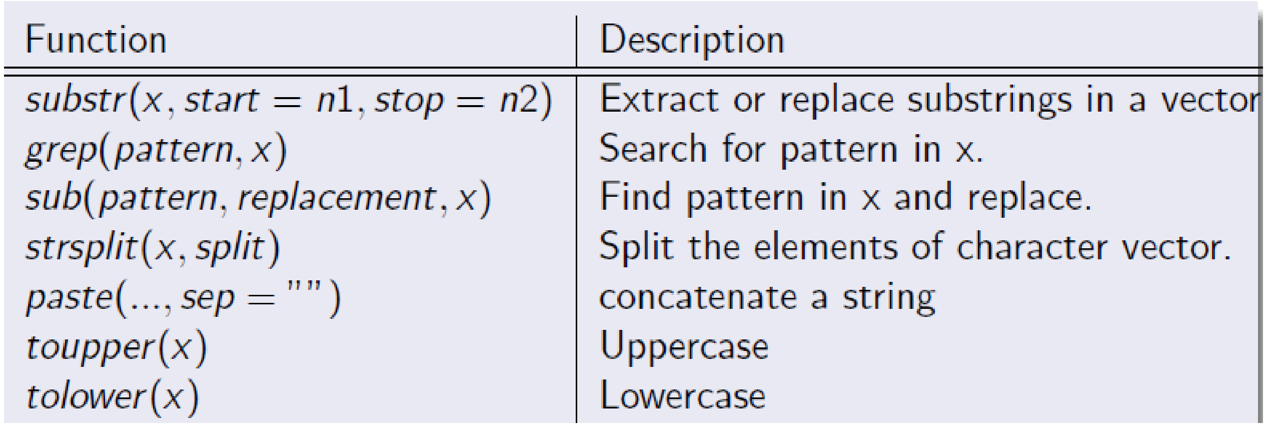


Function	Description
$abs(x)$	absolute value
$sqrt(x)$	square root
$ceiling(x)$	ceiling of a variable
$floor(x)$	floor of a variable
$trunc(x)$	trunc of a variable
$round(x, digits = n)$	round of a variable
$signif(x, digits = n)$	significant digits of a variable
$cos(x), sin(x), etc.$	trigonometric functions
$log(x)$	natural logarithm
$log10(x)$	logarithm base 10
$exp(x)$	exponent

Examples

```
sqrt (2)

cos(pi)

exp (2)
```

text functions

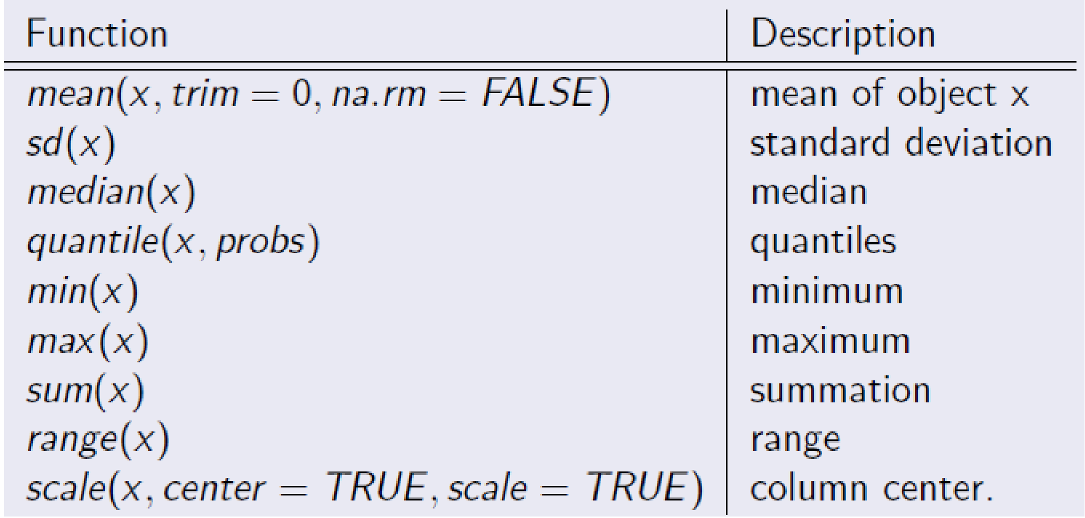


Function	Description
$substr(x, start = n1, stop = n2)$	Extract or replace substrings in a vector
$grep(pattern, x)$	Search for pattern in x.
$sub(pattern, replacement, x)$	Find pattern in x and replace.
$strsplit(x, split)$	Split the elements of character vector.
$paste(..., sep = "")$	concatenate a string
$toupper(x)$	Uppercase
$tolower(x)$	Lowercase

Examples

```
x <- " abcdef "

substr (x, 2, 4)

grep("A", c(" b","A","c "))

paste("x" ,1:3, sep ="")
```

statistical functions

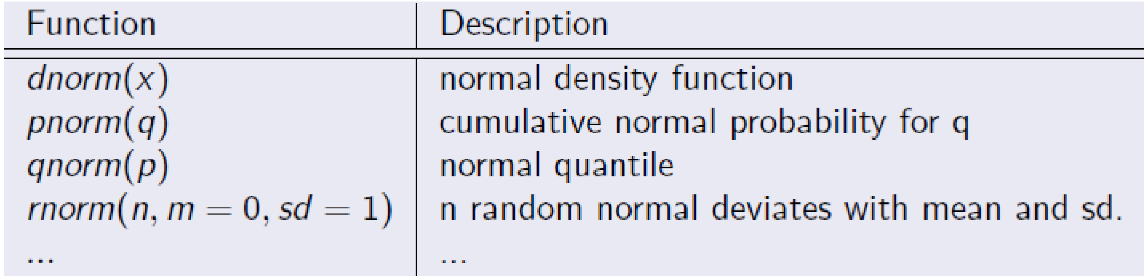


Function	Description
$mean(x, trim = 0, na.rm = FALSE)$	mean of object x
$sd(x)$	standard deviation
$median(x)$	median
$quantile(x, probs)$	quantiles
$min(x)$	minimum
$max(x)$	maximum
$sum(x)$	summation
$range(x)$	range
$scale(x, center = TRUE, scale = TRUE)$	column center.

Examples

```
x <- c(2 ,5 ,7)
mean (x)
max(x)
```

probability functions

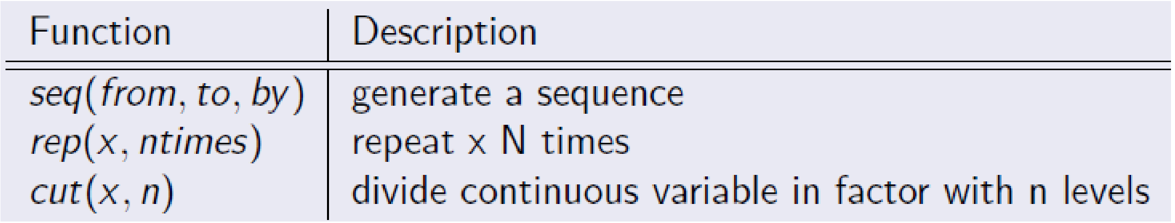


Function	Description
$dnorm(x)$	normal density function
$pnorm(q)$	cumulative normal probability for q
$qnorm(p)$	normal quantile
$rnorm(n, m = 0, sd = 1)$	n random normal deviates with mean and sd.
...	...

Examples

```
x <- rnorm (50 , m=50 , sd =10)
pnorm (1.96)
```

Useful functions



Function	Description
$seq(from, to, by)$	generate a sequence
$rep(x, ntimes)$	repeat x N times
$cut(x, n)$	divide continuous variable in factor with n levels

Examples

```
x <- seq (1 , 10 , 2)
y < - rep (1:3 , 2)
```

UPT

UNIVERSIDADE
PORTUCALENSE

Do conhecimento à prática.