Types and basic structures data in R

The purpose of the lecture is to familiarize yourself with the basic data types used in the R language, as well as with the basic structures that the R language operates on.

As a result of studying the lecture materials, you will know how to create data of various types, as well as operate on the main data structures.

Lecture questions

1. Data types in R
2. Basic data structures:

2.1 Vectors

2.2 Matrices

2.3 Arrays

2.4 Frames

2.5 Factors

2.6 Lists

Literary source :

1. Visual statistics. We use R! A. B. Shipunov, E. M. Baldin, P. A. Volkova, A. I. Korobeinikov, S. A. Nazarova, S. V. Petrov, V. G. Sufiyanov. 2014 year
2. Introduction to R: Notes on R: a programming environment for analyzing data and graphics. Version 3.1.0 (2014-04-10) U.N. Venables, D.M. Smith., Translation from English. - Moscow, 2014.109 s. - (series of technical documentation).
3. Statistical analysis and data visualization using R. S.E. Mastitsky, V.K. Shitikov, Heidelberg - London - Tolyatti, 2014.401 p. Website: http://r-analytics.blogspot.co Website: http://www.qsar4u.com/files/rintro/01.html

2. Data Types in R

- ✔ Structured and unstructured
- ✔ Clean and dirty
- ✔ Numerical, classification
- ✔ Symbols, text, pictures, speech
- ✔ 80% of the work is collecting and cleaning data !

Big data is usually BIG and unstructured

2. Data Types in R

The main data types	Description	Examples of values
numeric	integer objects (**integer**) Real numbers (**double**)	0L, 1L 0.1
logical	Logical objects: FALSE (F) , TRUE (T)	TRUE, FALSE or T, F
character (factor)	symbolic objects (variable values are specified in double or single quotes)	"hello, world!!!"
complex	numbers consisting of real and imaginary parts quotation marks)	3+4i
NA	Not available - missing data	Missing Values
NaN		NaN

2. Data Types in R

- Retrieving Data Type Information :

>class (x)

- Type Verification :

>is.[type] (x)

>is.list(x)

- Type cast :

>as.[type] (x)

>as. numeric(x)

```
>class(present$year)
[1] "numeric"

>is.logical(present$year)
[1] FALSE

>as.factor(present$year)
[1] 1940 1941 1942 1943 1944 1945 1946 1947 1948 1949 1950 1951 1952 1953
1954 [16] 1955 1956 1957 1958 1959 1960 1961 1962 1963 1964 1965 1966
1967 1968 1969 [31] 1970 1971 1972 1973 1974 1975 1976 1977 1978 1979
1980 1981 1982 1983 1984 [46] 1985 1986 1987 1988 1989 1990 1991 1992
1993 1994 1995 1996 1997 1998 1999 [61] 2000 2001 2002
63 Levels: 1940 1941 1942 1943 1944 1945 1946 1947 1948 1949 1950 1951
... 2002
```

2. Data Types in R

MISSING VALUES - NA

Often it is not possible to collect all the data on all parameters of the objects of interest to us.

Missing (unknown) observation values are indicated in R as NA (“Not available”).

NA test::

```
>is.na (x)
```

Getting rid of NA::

```
>na.omit (data)
```

2. Data Types in R

Define the data types for the columns of this table:

Таблица . Набор данных о пациентах

PatientID	AdmDate	Age	Diabetes	Status
1	10/15/2009	25	Type1	Poor
2	11/01/2009	34	Type2	Improved
3	10/21/2009	– ← Na !!!	Type1	Excellent
4	10/28/2009	52	Type1	Poor

Integer — discrete

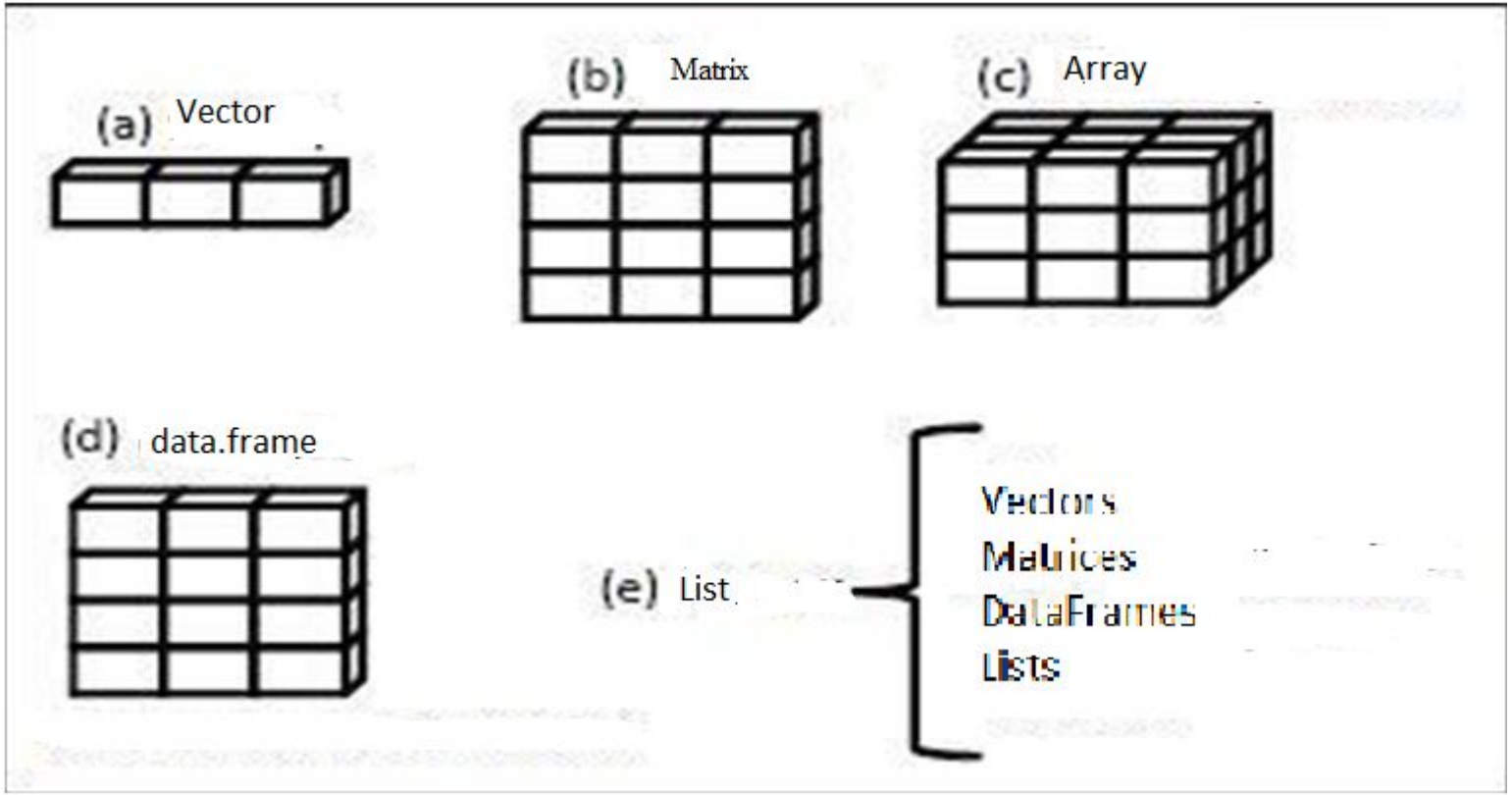
date

numeric — continuous

factor — nominal

factor — ordered

3. Basic data structures



3. Basic data structures

Data structure	Possible data types	Examples	Uniformity
vector	numeric, symbolic, complex, logical	c(1L, 2L, 3L) 1:3 vector("integer", 3)	homogeneous
Factor	numeric, character	factor(c("Male", "Female", "Male", "Male"))	
matrix (special case of an array *k* = 2)	numeric, symbolic, complex, logical	matrix(1:6, nrow = 2, ncol = 3)	

3. Basic data structures

Data structure	Possible data types	Examples	Uniformity
list	numeric, symbolic, complex, logical	list(1L, 2.3, "hi", F)	heterogeneous
data.frame	numeric, symbolic, complex, logical	data.frame(age = 18:23, height = c(170, 171, NA, 176, 173, 180), sex = factor(c("m", "f", "m", "m", "f", "m"))	
array	numeric, symbolic, complex, logical	array - table with k dimensions	

3. Features of the data structure in R

an R object is everything that can be represented in the form of variables, including constants, various data types, functions, and even diagrams.

Objects have: view (determines in what form the object is stored in memory) and a class (which tells common functions of type print how to handle it).

A data frame is a type of data structure in R that is similar to the type in which data is stored in ordinary statistical programs (in SAS, SPSS and STATA).

Columns are variables, and rows are observations. Variable types of variables can be contained in one data table. Data tables are the main type of data structure.

Factors are nominal or ordinal variables. In R, they are stored and processed in a special way.

3. Basic data structures: vectors

Vectors are vector data arrays that can contain numeric, textual, or logical data. To create a vector, the union function c () is used.:

```
a <- c(1, 2, 5, 3, 6, -2, 4)
b <- c("one", "two", "three")
c <- c(TRUE, TRUE, TRUE, FALSE, TRUE, FALSE)
```

3. Basic data structures: vectors

Individual elements of a vector can be called using a numerical vector consisting of element numbers in square brackets. For example, a [c (2, 4)] denotes the second and fourth elements of the vector.

```
a <- c(1, 2, 5, 3, 6, -2, 4)
a[3]
[1] 5

a[c(1, 3, 5)] [1] 1 5 6
a[2:6]
2  5  3  6 -2
```

The colon in the last example is used to create a sequence of numbers..
a <- c(2:6) is the same as a <- c(2, 3, 4, 5, 6).

3. Basic data structures: matrices

A matrix is a two-dimensional data array in which each element has the same type (numeric, textual, or logical). Common format :

mymatrix <- matrix(*vector*, nrow=*number_of_rows*, ncol=*number_of_columns*,
byrow=*logical_value*, dimnames=list(
char_vector_rownames, *char_vector_colnames*))

where **vector** contains elements of the matrix, **nrow** and **ncol** define the number of rows and columns in the matrix, and **dimnames** contains the names of rows and columns, which are stored as text vectors (they do not need to be specified). The **byrow** parameter determines whether the matrix should be filled by rows (byrow=TRUE) or by columns (by row=FALSE). By default, the matrix is populated by columns.

3. Basic data structures: matrices

Program code. Matrix Creation

```
y <- matrix(1:20, nrow=5, ncol=4)
 y
```

```
     [,1] [,2] [,3] [,4]
[1,]    1    6   11   16
[2,]    2    7   12   17
[3,]    3    8   13   18
[4,]    4    9   14   19
[5,]    5   10   15   20
```

3. Basic data structures: matrices

```
> cells      <- c(1,26,24,68)
>rnames  <-c("R1", "R2")
>cnames  <-c("C1", "C2")
mymatrix <- matrix(cells, nrow=2, ncol=2, byrow=TRUE, dimnames=list(rnames, cnames))
mymatrix              # 2 × 2 table filled in rows
      C1 C2
R1    1   26
R2    24  68
> mymatrix <- matrix(cells, nrow=2, ncol=2, byrow=FALSE, dimnames=list(rnames, cnames))

> mymatrix      # 2 × 2 table filled in columns
      C1 C2
R1    1 24
R2    26 68
```

3. Basic data structures: matrices

Using indexes when working with matrices

```
> x <- matrix(1:10, nrow=2)
> x
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    3    5    7    9
[2,]    2    4    6    8   10
> x[2,]     #  display the 2nd row of the matrix
[1]  2  4  6  8 10
> x[,2]     #  display the 2nd column of the matrix
[1] 3 4
> x[1,4]     # derive a matrix element from the 1st row and 4th column
[1] 7

> x[1, c(4,5)] # to display the matrix elements of the 1st row, 4-th and 5-th column
[1] 7 9
```

3. Basic data structures: arrays

Arrays are similar to matrices, but can have more than two dimensions.

myarray <- array(*vector*, *dimensions*, *dimnames*)

where vector contains the data itself, dimensions is a numeric vector specifying the dimension for each dimension and dimnames is an optional list of dimension names.

As an example, we give the program code, with the help of which a three-dimensional (2×3×4) array of numbers is created.

3. Basic data structures: arrays

```
>dim1 <- c("A1", "A2")
>dim2 <- c("B1", "B2", "B3")
>dim3 <- c("C1", "C2", "C3", "C4")
>z <- array(1:24, c(2, 3, 4), dimnames=list(dim1, dim2, dim3))
>z
, , C1
   B1 B2 B3
A1  1  3  5
A2  2  4  6
, , C2
   B1 B2 B3
A1  7  9 11
A2  8 10 12
, , C3
   B1 B2 B3
A1 13 15 17
A2 14 16 18
, , C4
   B1 B2 B3
A1 19 21 23
A2 20 22 24
```

3. Basic data structures: dataframes

A data frame is a more widely used object than a matrix because different columns can contain different types of data (numeric, text, etc.). A data table is the most commonly used data structure in R.

A set of data about patients (table. above) consists of numeric and textual data. This data needs to be represented as a data table, not a matrix, because there are different types of data here.The data table is created using the data function.frame():

mydata < - data.frame(col1, col2, col3,…),

where-col1, col2, col3,... are vectors of any type (textual, numeric, or logical) that will become table columns. Names can be assigned to each column using the names () function. Let's illustrate this with an example of the program code.

3. Basic data structures: dataframes

```
patientID <- c(1, 2, 3, 4)
age <- c(25, 34, 28, 52)
diabetes <- c("Type1", "Type2", "Type1", "Type1")
status <- c("Poor", "Improved", "Excellent", "Poor")

patientdata <- data.frame(patientID, age, diabetes, status)
patientdata
  patientID age diabetes    status
1         1  25    Type1      Poor
2         2  34    Type2  Improved
3         3  28    Type1 Excellent
4         4  52    Type1      Poor
```

3. Basic data structures: dataframes

Designation of data table elements

```
>patientdata[1:2]
patientID   age
1          25
2          34
3          28
4           52

> patientdata[c("diabetes", "status")]
       diabetes  status
1      Type1     Poor
2      Type2     Improved
3      Type1     Excellent

patientdata$age [1] 25 34 28 52
```

3. Basic data structures: factors

The factor () function stores categorical data as a vector of integers in the range from one to k (where k is the number of unique values of the categorical variable) and as an internal vector of a chain of characters (the original values of the variable) corresponding to these integers.

```
diabetes <- c("Type1", "Type2", "Type1", "Type1").

diabetes <- factor(diabetes)
```

Numeric values are assigned in alphabetical order. Any analysis you do with the diabetes vector will take this variable as nominal and choose statistical methods that are appropriate for this type of data.

3. Basic data structures: factors

You can change the default setting by specifying the levels parameter. For example:

```
>status <- factor(status, order=TRUE,

levels=c("Poor", "Improved", "Excellent"))
```

will assign levels to the values of the vector as follows:

1=Poor, 2=Improved, 3=Excellent.

3. Basic data structures: factors

The use of factors

```
>patientID <- c(1, 2, 3, 4)        # Enter the data as vectors
>age <- c(25, 34, 28, 52)

diabetes <- c("Type1", "Type2", "Type1", "Type1")
status <- c("Poor", "Improved", "Excellent", "Poor")
diabetes <- factor(diabetes)           # we point out that diabetes is a factor
status <- factor(status, order=TRUE)      # status – it is an ordered factor
>patientdata <- data.frame(patientID, age, diabetes, status) # combine the data into a table

> str(patientdata)
```

3. Basic data structures: lists

Lists are the most complex data type in R. In fact, a list is an ordered list of objects (components). For example, a list can be a combination of vectors, matrices, data tables, and even other lists. The list can be created using the function list():

mylist <- list(*object1*, *object2*, …),

where objects are any data structures we discussed before. Objects in the list can be named:

mylist <- list(*name1=object1*, *name2=object2*, …)

3. Basic data structures: lists

Creating a list

```
>g <- "My First List"
>h <- c(25, 26, 18, 39)

>j <- matrix(1:10, nrow=5)
>k <- c("one", "two", "three")

> mylist <- list(title=g, ages=h, j, k)

 > mylist              # Display the entire list
   mylist[[2]]         # Display the second object of the list
> mylist[["ages"]]     # Display the second object of the list
```

Conclusions of the lecture

We learned:

- What data types are used in R
- What objects does R operate on
- Features of working with basic R structures

- Apply arithmetic operators to variables and vectors
- To calculate some statistics with the use of aggregate functions