Indexing data frames and vectors

Data tameRs

 Press `A` to change slides into text Press `T` to display table of contents.

What is this episode about?

We will work most of the time on data in the form of tables or vectors. One of the basic operations performed on tables and vectors is selecting subsets of rows, columns or values.

In this episode you will learn:

- how to create vectors,
- how to index values from vectors/arrays,
- how to select rows from data frames,
- how to select columns from data frames,
- how to select rows and columns,
- how to index rows and columns using names and logical values.

Vectors

One of the standard types of data in R are vectors.

Vectors may consist of numbers, words, logical values or other types of values. Let us start from numerical values.

In R even one value constitutes a vector, although it is a small and one-element vector.

```
4
```

```
## [1] 4
```

Longer vectors can be created with the function c(). It allows us to make one vector out of several values. Below you can see an example command creating a vector composed of three elements.

```
c(3, 4, 5)
```

```
## [1] 3 4 5
```

Programming and data analysis frequently uses vectors of

Vectors

The c() function allows you to create vectors of logical values (with two optional states labeled TRUE/FALSE)), text or other types of values.

If you want to use the vector in the future, you need to attribute it to a variable. You may perform that operation using the operator <- or =. If you want to display the content of the variable, you only need to enter its name into the console and press ENTER.

```
co_drugi <- c(TRUE, FALSE, TRUE, FALSE, TRUE, FALSE)
co_drugi
```

```
## [1]  TRUE FALSE  TRUE FALSE  TRUE FALSE
```

```
literki <- c("alfa", "beta", "gamma", "delta")
literki
```

```
## [1] "alfa"  "beta"  "gamma" "delta"
```

Indexing vectors

Indexing means selecting specific values. We select values by referring to their indices.

A vector is a sequence of elements. The first element of this sequence has an index of 1, the next one has an index of 2, the next of 3 and so on up to the last element. Index of the last element is also the length of the vector. It can be checked by the function `length()`.

```
length(LETTERS)
```

```
## [1] 26
```

If you want to refer to specific indices of the vector, use the operator `[]`. Write the index of the element which you want to refer to inside the square brackets.

```
# first element of the vector
LETTERS[1]
```

Indexing vectors

When you refer to vectors you may write indices of more than one value. Remember only that indices must be vectors. For this purpose you will need the `c()` function discussed above.

For example, in order to select the first, fifth and last element of the LETTERS vector you first need to create a vector with three indices.

```
LETTERS[c(1,5,26)]
```

```
## [1] "A" "E" "Z"
```

There is another equally correct solution. You can first create a vector of indices and then use is to index the `LETTERS` vector.

```
indeksy <- c(1, 5, 26)
LETTERS[indeksy]
```

```
## [1] "A" "E" "Z"
```

Sequences are very useful when you want to select more elements

Indexing vectors

Sequences of values do not necessarily need to be increasing. They can be used to reverse the order of the elements of the vector, for example:

```
10:1
```

```
##  [1] 10  9  8  7  6  5  4  3  2  1
```

```
LETTERS[10:1]
```

```
##  [1] "J" "I" "H" "G" "F" "E" "D" "C" "B" "A"
```

Elements of a vector can also be indexed with a logical expression.

In the example below the command `LETTERS > "K"` creates a vector of logical values which determines whether the following letter is bigger or smaller than `K` (in the lexicographic order). Such vector of logical values can be used in indexing the `LETTERS` vector.

```
LETTERS > "K"
```

Exercises

- Build a sequence of ten subsequent letters of the Latin alphabet.
- Build a sequence of ten subsequent odd numbers starting from 3.
- Select letters with indices 5, 10, 15, 20 and 25 from the vector LETTERS.
- Write values of the vector LETTERS backwards.

Cats vs. birds

We will use a small data set about cats and dogs to learn selecting rows and columns.

This data set is available in the package `PogromcyDanych` ?just load the package. Other ways in which you can load that particular data set were presented in the episode 5.

Seven columns and thirteen rows is just the right data set to excercise.

```
library(PogromcyDanych)
koty_ptaki
```

```
##          gatunek   waga dlugosc predkosc  habitat zywot
## 1         Tygrys 300.00     2.5        60     Azja
## 2            Lew 200.00     2.0        80   Afryka
## 3         Jaguar 100.00     1.7        90  Ameryka
## 4           Puma  80.00     1.7        70  Ameryka
## 5        Leopard  70.00     1.4        85     Azja
## 6         Gepard  60.00     1.4       115   Afryka
```

Indexing rows in the data frame

The principal difference between any vector and a data frame is the fact that vectors are one-dimensional. Their values form sequences. A data frame, on the other hand, presents data located in two dimensions of rows and columns. When we index values in a data frame we give a vector of indices for rows and vector of values for columns.

You can use the operator `[,]` to refer to rows or columns of the data frame. A comma is an indispensable element. Write indices for rows before the comma and indices for columns after the comma. If there is no value before or after comma, the program will select all the elements of the column / row.

This is an example of reference to the third row of the data frame.

```
koty_ptaki[3, ]
```

```
##   gatunek waga dlugosc predkosc habitat zywotnosc druzyn
## 3  Jaguar  100     1.7        90 Ameryka        15      Ko
```

Indexing rows in the data frame

If you want to choose more than one row you need to write their indices separating them with commas, just like in case of vectors.

In order to refer to several subsequent rows you can use a sequence created with the operator `:`.

For example, rows from 8 to 10 of the data frame `koty_ptaki` can be selected with the following command.

```
koty_ptaki[8:10, ]
```

```
##           gatunek   waga dlugosc predkosc  habitat zywotn
## 8          Jerzyk   0.05     0.2       170 Euroazja
## 9           Strus 150.00     2.5        70   Afryka
## 10 Orzel przedni   5.00     0.9       160   Polnoc
```

The function `c()` connects values and sequences and transforms them into a vector which can be used in indexing rows.

The following command selects rows number 3, 8, 9 and 10.

Indexing rows in the data frame

Vector of the logical values can also be used as index.

Looking ahead a little bit, we will use the column `predkosc` to choose only these rows of the data frame for which the value of speed does not exceed 100.

```
fastest <- koty_ptaki$predkosc > 100
fastest
```

```
##  [1] FALSE FALSE FALSE FALSE FALSE  TRUE FALSE  TRUE FAI
## [12] FALSE  TRUE
```

```
koty_ptaki[fastest, ]
```

```
##              gatunek  waga dlugosc predkosc  habitat zywotn
## 6            Gepard 60.00     1.4      115   Afryka
## 8            Jerzyk  0.05     0.2      170 Euroazja
## 10  Orzel przedni  5.00     0.9      160   Polnoc
## 11 Sokol wedrowny  0.70     0.5      110   Polnoc
```

Indexing rows in the data frame

Negative indices can also be used in indexing rows and columns. These indices refer to all the values except those specifically selected.

For example, all the rows apart from rows no 1,3,8,9 and 10 can be selected with this command.

```
koty_ptaki[ -c(1, 3, 8:10), ]
```

```
##              gatunek  waga dlugosc predkosc  habitat zywotn
## 2                Lew 200.0     2.0       80   Afryka
## 4               Puma  80.0     1.7       70  Ameryka
## 5            Leopard  70.0     1.4       85     Azja
## 6             Gepard  60.0     1.4      115   Afryka
## 7              Irbis  50.0     1.3       65     Azja
## 11 Sokol wedrowny   0.7     0.5      110   Polnoc
## 12 Sokol norweski   2.0     0.7      100   Polnoc
## 13         Albatros   4.0     0.8      120 Poludnie
```

Indexing columns in the data frame

Columns can be indexed just like rows.

In order to select the second column you may write its number after comma.

```
koty_ptaki[, 2]
```

```
##  [1] 300.00 200.00 100.00  80.00  70.00  60.00  50.00
## [11]   0.70   2.00   4.00
```

When you select one column your result is a vector, not a data frame. You can easily imagine this when you take a look at the manner in which the data is displayed.

If you do not want to transform a column into vector and still receive a data frame, you need to add an argument `drop=FALSE` to the indexing operator.

```
koty_ptaki[,2, drop=FALSE]
```

Indexing columns in the data frame

Columns in the data frame can be indexed not only with numbers but also with names (columns bear names).

Names of the columns in the data frame can be read with the function `colnames()`. When you use this function you will receive a vector with names of columns.

```
colnames(koty_ptaki)
```

```
## [1] "gatunek"   "waga"      "dlugosc"   "predkosc"   "hab
## [7] "druzyna"
```

If you want to select a column entitled `waga` from the data frame, you may use that name as an index.

```
koty_ptaki[, "waga"]
```

```
##  [1] 300.00 200.00 100.00  80.00  70.00  60.00  50.00
## [11]   0.70   2.00   4.00
```

Indexing columns in the data frame

If you want to select more than one column, you can use the function `c()`, just like in case of rows and vectors.

For example, if you want to select the second, fourth, fifth and sixth column, you can use the following command.

```
# we may just as well write:
# koty_ptaki[, c("waga", "predkosc", "habitat", "zywotnosc
koty_ptaki[, c(2,4:6)]
```

```
##      waga predkosc  habitat zywotnosc
## 1  300.00       60     Azja        25
## 2  200.00       80   Afryka        29
## 3  100.00       90  Ameryka        15
## 4   80.00       70  Ameryka        13
## 5   70.00       85     Azja        21
## 6   60.00      115   Afryka        12
## 7   50.00       65     Azja        18
## 8    0.05      170 Euroazja        20
```

Selecting sub-frames from the data frame

We can refer to both rows and columns in the data frame at the same time by selecting one of its sub-frames.

For example, you may select four rows and four columns with the following command.

```
koty_ptaki[c(3,8:10), c(2,4:6)]
```

```
##      waga predkosc  habitat zywotnosc
## 3  100.00       90  Ameryka        15
## 8    0.05      170 Euroazja        20
## 9  150.00       70   Afryka        45
## 10   5.00      160   Polnoc        20
```

Selection of four rows and one column.

```
koty_ptaki[c(3,8:10), 2]
```

```
## [1] 100.00   0.05 150.00   5.00
```

Selecting sub-frames from data frames with the use of names

You already know how to refer to columns by using their names.

You can do the same with rows. The function `rownames()` reveals names of rows in the data frame.

```
rownames(koty_ptaki)
```

```
##   [1] "1"   "2"   "3"   "4"   "5"   "6"   "7"   "8"   "9"   "10"  "
```

Yet these names say very little and it would not make sense to use them in indexing. Let us then change the names of rows for the names located in the column `gatunek`.

```
rownames(koty_ptaki) <- koty_ptaki$gatunek
```

Now we can refer to the rows through names of these rows.

In the following example rows referring to the four chosen species and three columns are selected. If names of rows are meaningful.

Sorting through indexing

I would like to present a very interesting and a little advanced use of indexing on the example of the function `order()`.

This function creates indexes of subsequent growing values.

For example, the column `predkosc` contains the following values.

```
koty_ptaki[,"predkosc"]
```

```
##  [1]  60  80  90  70  85 115  65 170  70 160 110 100 120
```

The function `order()` creates indexes of subsequent growing values. The smallest value is 60 and it?s in the first position. The next value is 65 and it occupies the 7th position. Then we have 70 ?at the 4th and 9th positions and so forth.

```
order(koty_ptaki[,"predkosc"])
```

```
##  [1]  1  7  4  9  2  5  3 12 11  6 13 10  8
```

Summary of R commands

This episode was devoted to functions creating vectors and sequences as well as functions indexing vectors.

```r
# creating vectors of numbers and sequences
c(3, 4, 5)
2:7
seq(from = 3, to = 15, by = 2)

# logical and character vector
co_drugi <- c(TRUE, FALSE, TRUE, FALSE, TRUE, FALSE)
co_drugi

literki <- c("alfa", "beta", "gamma", "delta")
literki

# determining the length of a vector
length(LETTERS)

# indexing a single element (in this case: the first one)
```

Summary of R commands

We have discussed functions allowing us to index rows in the data frames.

```
# indexing a single row
koty_ptaki[3, ]

# number of rows
nrow(koty_ptaki)

# indexing several rows
koty_ptaki[8:10, ]
koty_ptaki[c(3, 8:10), ]

# using auxiliary variables
indeksy <- c(3, 8:10)
koty_ptaki[indeksy, ]

# first 6 rows, last 6 rows
head(koty_ptaki)
```

Summary of R commands

We have discussed functions allowing us to index rows and columns in the data frame separately and jointly.

```
# indexing columns. In this case: second column as a vecto
koty_ptaki[, 2]
# and as a data.frame, using argument drop=FALSE
koty_ptaki[,2, drop=FALSE]

# column names
colnames(koty_ptaki)

# indexing the single column using its name
koty_ptaki[, "waga"]
koty_ptaki$waga

# indexing multiple columns
# koty_ptaki[, c("waga", "predkosc", "habitat", "zywotnosc
koty_ptaki[, c(2,4:6)]
```

Exercises

- Select all the rows except “Soko?y” (falcons: “Sokol wedrowny” and “Sokol norweski”, rows 11 and 12) from the data frame `koty_ptaki`.
- Select only cats (first seven rows) from the data frame `koty_ptaki`.
- Select only the column with weight and speed from the data frame `koty_ptaki`.
- Select all the columns the data frame `koty_ptaki` except the last one.
- Select rows for which weight is less than 100 and first four columns from `koty_ptaki`.

You can find sample answers at `https://rawgit.com/pbiecek/MOOC/master/0_dane/9_zadania.html`