Indexing data frames and vectors

Data Crunchers

season 1 /episode 7

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 selection of 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() for example. It allows 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 subsequent numerical values. They are referred to as sequences.

You may use the operator : to create sequences of subsequent numbers. However, if you want to build a sequence of numbers with even intervals other than 1, you may use a more convenient function seq().

**2:7**

**## [1] 2 3 4 5 6 7**

**seq(from = 3, to = 15, by = 2)**

**## [1] 3 5 7 9 11 13 15**

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"**

In the next part of this episode we will use a vector of subsequent letters of the English alphabet saved in the variable LETTERS. We need it for our training in indexing.

**LETTERS**

**## [1] "A" "B" "C" "D" "E" "F" "G" "H" "I" "J" "K" "L" "M" "N" "O" "P" "Q" ## [18] "R" "S" "T" "U" "V" "W" "X" "Y" "Z"**

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().

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.

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.

**length(LETTERS)**

**## [1] 26**

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 create first 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 from a vector.

***# pierwszy element wektora* LETTERS[1]**

**## [1] "A"**

***# piąty* LETTERS[5]**

**## [1] "E"**

***# ostatni element wektora* LETTERS[26]**

**## [1] "Z"**

***# zamiast podawać wartość 26 można wstawić funkcję, która jako wynik zwróci długość wektora* LETTERS[length(LETTERS)]**

**## [1] "Z"**

***# dziesięć pierwszych liter* LETTERS[1:10]**

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

***# pięć pierwszych i pięć ostatnich* LETTERS[c(1:5, 21:26)]**

**## [1] "A" "B" "C" "D" "E" "U" "V" "W" "X" "Y" "Z"**

***# co druga litera, indeksujemy sekwencją od 1 do 26 z krokiem co 2* coDruga <- LETTERS[seq(from = 1, to = 26, by = 2)] coDruga**

**## [1] "A" "C" "E" "G" "I" "K" "M" "O" "Q" "S" "U" "W" "Y"**

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 condition.

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"**

**## [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE ## [12] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE ## [23] TRUE TRUE TRUE TRUE**

***# tylko litery spełniające określony warunek* *# w tym przypadku litery występujące po literze 'K'* LETTERS[LETTERS > "K"]**

**## [1] "L" "M" "N" "O" "P" "Q" "R" "S" "T" "U" "V" "W" "X" "Y" "Z"**

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 for training.

**library(PogromcyDanych) koty\_ptaki**

**## gatunek waga dlugosc predkosc habitat zywotnosc druzyna ## 1 Tygrys 300.00 2.5 60 Azja 25 Kot ## 2 Lew 200.00 2.0 80 Afryka 29 Kot ## 3 Jaguar 100.00 1.7 90 Ameryka 15 Kot ## 4 Puma 80.00 1.7 70 Ameryka 13 Kot ## 5 Leopard 70.00 1.4 85 Azja 21 Kot ## 6 Gepard 60.00 1.4 115 Afryka 12 Kot ## 7 Irbis 50.00 1.3 65 Azja 18 Kot ## 8 Jerzyk 0.05 0.2 170 Euroazja 20 Ptak ## 9 Strus 150.00 2.5 70 Afryka 45 Ptak ## 10 Orzel przedni 5.00 0.9 160 Polnoc 20 Ptak ## 11 Sokol wedrowny 0.70 0.5 110 Polnoc 15 Ptak ## 12 Sokol norweski 2.00 0.7 100 Polnoc 20 Ptak ## 13 Albatros 4.00 0.8 120 Poludnie 50 Ptak**

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.

**koty\_ptaki[3, ]**

**## gatunek waga dlugosc predkosc habitat zywotnosc druzyna ## 3 Jaguar 100 1.7 90 Ameryka 15 Kot**

You can use the operator [,] to refer to rows or columns of the data frame. A coma is an indispensable element. Write indices for rows before the coma and indices for columns after the coma. If there is no value before or after coma, 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.

You can check the number of rows using the function nrow() (we check the number of columns with the function ncol() respectively).

These functions are very useful if you want to display the last row for example.

**nrow(koty\_ptaki)**

**## [1] 13**

**koty\_ptaki[13, ]**

**## gatunek waga dlugosc predkosc habitat zywotnosc druzyna ## 13 Albatros 4 0.8 120 Poludnie 50 Ptak**

Indexing rows in the data frame

If you want to choose more than one row you need to write the their indices separating them with comas, 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\_patki can be selected with the following command.

**koty\_ptaki[8:10, ]**

**## gatunek waga dlugosc predkosc habitat zywotnosc druzyna ## 8 Jerzyk 0.05 0.2 170 Euroazja 20 Ptak ## 9 Strus 150.00 2.5 70 Afryka 45 Ptak ## 10 Orzel przedni 5.00 0.9 160 Polnoc 20 Ptak**

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.

**koty\_ptaki[c(3, 8:10), ]**

**## gatunek waga dlugosc predkosc habitat zywotnosc druzyna ## 3 Jaguar 100.00 1.7 90 Ameryka 15 Kot ## 8 Jerzyk 0.05 0.2 170 Euroazja 20 Ptak ## 9 Strus 150.00 2.5 70 Afryka 45 Ptak ## 10 Orzel przedni 5.00 0.9 160 Polnoc 20 Ptak**

Or to the same effect:

**indeksy <- c(3, 8:10) koty\_ptaki[indeksy, ]**

**## gatunek waga dlugosc predkosc habitat zywotnosc druzyna ## 3 Jaguar 100.00 1.7 90 Ameryka 15 Kot ## 8 Jerzyk 0.05 0.2 170 Euroazja 20 Ptak ## 9 Strus 150.00 2.5 70 Afryka 45 Ptak ## 10 Orzel przedni 5.00 0.9 160 Polnoc 20 Ptak**

Functions head() and tail() serve to select several first or last rows of the data set. Both functions can be developed by an addition of a second argument specifying number of first/last rows that we want to read –the default number is 6.

**head(koty\_ptaki)**

**## gatunek waga dlugosc predkosc habitat zywotnosc druzyna ## 1 Tygrys 300 2.5 60 Azja 25 Kot ## 2 Lew 200 2.0 80 Afryka 29 Kot ## 3 Jaguar 100 1.7 90 Ameryka 15 Kot ## 4 Puma 80 1.7 70 Ameryka 13 Kot ## 5 Leopard 70 1.4 85 Azja 21 Kot ## 6 Gepard 60 1.4 115 Afryka 12 Kot**

**tail(koty\_ptaki)**

**## gatunek waga dlugosc predkosc habitat zywotnosc druzyna ## 8 Jerzyk 0.05 0.2 170 Euroazja 20 Ptak ## 9 Strus 150.00 2.5 70 Afryka 45 Ptak ## 10 Orzel przedni 5.00 0.9 160 Polnoc 20 Ptak ## 11 Sokol wedrowny 0.70 0.5 110 Polnoc 15 Ptak ## 12 Sokol norweski 2.00 0.7 100 Polnoc 20 Ptak ## 13 Albatros 4.00 0.8 120 Poludnie 50 Ptak**

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 from the data frame only these rows for which the value of speed does not exceed 100.

**najszybsze <- koty\_ptaki$predkosc > 100 najszybsze**

**## [1] FALSE FALSE FALSE FALSE FALSE TRUE FALSE TRUE FALSE TRUE TRUE ## [12] FALSE TRUE**

**koty\_ptaki[najszybsze, ]**

**## gatunek waga dlugosc predkosc habitat zywotnosc druzyna ## 6 Gepard 60.00 1.4 115 Afryka 12 Kot ## 8 Jerzyk 0.05 0.2 170 Euroazja 20 Ptak ## 10 Orzel przedni 5.00 0.9 160 Polnoc 20 Ptak ## 11 Sokol wedrowny 0.70 0.5 110 Polnoc 15 Ptak ## 13 Albatros 4.00 0.8 120 Poludnie 50 Ptak**

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 zywotnosc druzyna ## 2 Lew 200.0 2.0 80 Afryka 29 Kot ## 4 Puma 80.0 1.7 70 Ameryka 13 Kot ## 5 Leopard 70.0 1.4 85 Azja 21 Kot ## 6 Gepard 60.0 1.4 115 Afryka 12 Kot ## 7 Irbis 50.0 1.3 65 Azja 18 Kot ## 11 Sokol wedrowny 0.7 0.5 110 Polnoc 15 Ptak ## 12 Sokol norweski 2.0 0.7 100 Polnoc 20 Ptak ## 13 Albatros 4.0 0.8 120 Poludnie 50 Ptak**

Notice that in this example the sign – located before the function c() creating a vector modifies the sign of all the elements of that vector.

**-c(1, 3, 8:10)**

**## [1] -1 -3 -8 -9 -10**

Note! You shouldn’t to use simultaneously positive and negative signs.

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 coma.

**koty\_ptaki[, 2]**

**## [1] 300.00 200.00 100.00 80.00 70.00 60.00 50.00 0.05 150.00 5.00 ## [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]**

**## waga ## 1 300.00 ## 2 200.00 ## 3 100.00 ## 4 80.00 ## 5 70.00 ## 6 60.00 ## 7 50.00 ## 8 0.05 ## 9 150.00 ## 10 5.00 ## 11 0.70 ## 12 2.00 ## 13 4.00**

Note. This strange form of command results from the fact that the operator [,] is really a function. You will learn more about more advanced elements of the programming language in the next episodes of the course.

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" "habitat" "zywotnosc" ## [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 0.05 150.00 5.00 ## [11] 0.70 2.00 4.00**

We refer to only one column quite often while working on R so this operation can be performed in more than way. The most popular way is to use the operator $.

Using this operator we can refer to the columns named waga in the following way.

**koty\_ptaki$waga**

**## [1] 300.00 200.00 100.00 80.00 70.00 60.00 50.00 0.05 150.00 5.00 ## [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.

***# równoważnie moglibyśmy napisać*  *# 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 ## 9 150.00 70 Afryka 45 ## 10 5.00 160 Polnoc 20 ## 11 0.70 110 Polnoc 15 ## 12 2.00 100 Polnoc 20 ## 13 4.00 120 Poludnie 50**

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**

Selection of one row and four columns.

**koty\_ptaki[3, c(2,4:6)]**

**## waga predkosc habitat zywotnosc ## 3 100 90 Ameryka 15**

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" "11" "12" "13"**

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, they facilitate selecting the rows a great deal.

**koty\_ptaki[c("Lew", "Leopard", "Jerzyk", "Strus"), c("waga", "dlugosc", "predkosc")]**

**## waga dlugosc predkosc ## Lew 200.00 2.0 80 ## Leopard 70.00 1.4 85 ## Jerzyk 0.05 0.2 170 ## Strus 150.00 2.5 70**

Note. If you want to get rid of names of rows, you can do so with the following command.

**rownames(koty\_ptaki) <- NULL**

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().

**koty\_ptaki[,"predkosc"]**

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

This function creates indexes of subsequent growing values.

**order(koty\_ptaki[,"predkosc"])**

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

For example, the column predkosc contains the following values.

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.

We can use this result (błąd literówka po polsku) to sort the data frame by a certain column. An example below presents use of the function order() in creation of the vector kolejnosc (błąd literówka po polsku) which is in turn used in indexing the data frame called koty\_ptaki.

**kolejnosc <- order(koty\_ptaki[,"predkosc"]) koty\_ptaki[kolejnosc, ]**

**## gatunek waga dlugosc predkosc habitat zywotnosc ## Tygrys Tygrys 300.00 2.5 60 Azja 25 ## Irbis Irbis 50.00 1.3 65 Azja 18 ## Puma Puma 80.00 1.7 70 Ameryka 13 ## Strus Strus 150.00 2.5 70 Afryka 45 ## Lew Lew 200.00 2.0 80 Afryka 29 ## Leopard Leopard 70.00 1.4 85 Azja 21 ## Jaguar Jaguar 100.00 1.7 90 Ameryka 15 ## Sokol norweski Sokol norweski 2.00 0.7 100 Polnoc 20 ## Sokol wedrowny Sokol wedrowny 0.70 0.5 110 Polnoc 15 ## Gepard Gepard 60.00 1.4 115 Afryka 12 ## Albatros Albatros 4.00 0.8 120 Poludnie 50 ## Orzel przedni Orzel przedni 5.00 0.9 160 Polnoc 20 ## Jerzyk Jerzyk 0.05 0.2 170 Euroazja 20 ## druzyna ## Tygrys Kot ## Irbis Kot ## Puma Kot ## Strus Ptak ## Lew Kot ## Leopard Kot ## Jaguar Kot ## Sokol norweski Ptak ## Sokol wedrowny Ptak ## Gepard Kot ## Albatros Ptak ## Orzel przedni Ptak ## Jerzyk Ptak**

Summary of R commands

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

***# tworzenie wektorów liczb i sekwencji* c(3, 4, 5) 2:7 seq(from = 3, to = 15, by = 2) *# wektory wartości logicznych i napisowych* co\_drugi <- c(TRUE, FALSE, TRUE, FALSE, TRUE, FALSE) co\_drugi literki <- c("alfa", "beta", "gamma", "delta") literki *# wyznaczanie długości wektora LETTERS* length(LETTERS) *# indeksowanie pojedynczego elementu wektora, pierwszy element* LETTERS[1] *# zamiast podawać wartość 26 można wstawić funkcję, która jako wynik zwróci długość wektora* LETTERS[length(LETTERS)] *# indeksowanie kilku elementów wektora* LETTERS[c(1,5,26)] *# z użyciem zmiennej pomocniczej* indeksy <- c(1, 5, 26) LETTERS[indeksy] *# dziesięć pierwszych liter od J do A* LETTERS[10:1] *# pięć pierwszych i pięć ostatnich* LETTERS[c(1:5, 21:26)] *# co druga litera, indeksujemy sekwencją od 1 do 26 z krokiem co 2* LETTERS[seq(from = 1, to = 26, by = 2)] *# używanie wartości logicznych do indeksowania wektora tylko litery występujące po literze 'K'* LETTERS[LETTERS > "K"]**

Summary of R commands

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

***# indeksowanie pojedynczego wiersza* koty\_ptaki[3, ] *# liczba wierszy* nrow(koty\_ptaki) *# indeksowanie kilku wierszy*  koty\_ptaki[8:10, ] koty\_ptaki[c(3, 8:10), ] *# z użyciem zmiennej pomocniczej* indeksy <- c(3, 8:10) koty\_ptaki[indeksy, ] *# pierwsze 6 wierszy i ostatnie 6 wierszy* head(koty\_ptaki) tail(koty\_ptaki) *# indeksowanie wierszy warunkiem logicznym* najszybsze <- koty\_ptaki$predkosc > 100 koty\_ptaki[najszybsze, ] *# używanie ujemnych indeksów by pominąć wiersze* koty\_ptaki[ -c(1, 3, 8:10), ]**

Summary of R commands

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

***# indeksowanie kolumn, tylko druga kolumna, wynikiem jest wektor* koty\_ptaki[, 2] *# drop=FALSE powoduje, że wynikiem jest ramka danych* koty\_ptaki[,2, drop=FALSE] *# nazwy kolumn* colnames(koty\_ptaki) *# indeksowanie jednej kolumny z użyciem nazwy* koty\_ptaki[, "waga"] koty\_ptaki$waga *# indeksowanie kilku kolumn*  *# koty\_ptaki[, c("waga", "predkosc", "habitat", "zywotnosc")]* koty\_ptaki[, c(2,4:6)] *# jednoczesne indeksowanie wierszy i kolumn* koty\_ptaki[c(3,8:10), c(2,4:6)] *# wektor nazw wierszy* rownames(koty\_ptaki) *# przypisanie nowych nazw dla wierszy* rownames(koty\_ptaki) <- koty\_ptaki$gatunek *# odwoływanie się przez nazwy wierszy i kolumn* koty\_ptaki[c("Lew", "Leopard", "Jerzyk", "Strus"), c("waga", "dlugosc", "predkosc")]**

Exercises

* Select all the rows except ‘Sokoły’ (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 does not exceed 100 and four first columns from the data frame.

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