Hands-On Exercise 2

Elena Dubova

6/24/2020

Let's go!

Today we cover working with dataframes. And we will start with canonical data science dataset - **iris**.

Attribute Information:

1. sepal length in cm
2. sepal width in cm
3. petal length in cm
4. petal width in cm
5. class: – Iris Setosa – Iris Versicolour – Iris Virginica

Dimensions and Structure of the Dataframe

```
data <- iris
dim(data)
```

```
## [1] 150   5
```

Dataframe, as opposed to matrix can contain different types of variables.

```
str(data)
```

```
## 'data.frame':    150 obs. of  5 variables:
##  $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
##  $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
##  $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
##  $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
##  $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
```

And is, in essence, 2-dimensional structure of rows and columns. Every row is an observation, every column is a feature.

```
head(data)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
## 4          4.6         3.1          1.5         0.2  setosa
## 5          5.0         3.6          1.4         0.2  setosa
## 6          5.4         3.9          1.7         0.4  setosa
```

Compare it with 1 dimentional structure - a list

```
vector <- sample(1:100, 20, replace=FALSE)
vector
```

```
##  [1] 83 84 52  9 30 20 27 44 21 56 82 86 95 40 42  7 12 54 60 61
```

```
length(vector)
```

```
## [1] 20
```

Every column of a dataframe is, in fact, can be viewed as a vector.

```
#data$Sepal.Length
#length(data$Sepal.Length)
```

You can use a value or an index depending on the scenario. What is the maximum sepal length in the dataset?

```
max(data$Sepal.Length)
```

```
## [1] 7.9
```

```
which.max(data$Sepal.Length)
```

```
## [1] 132
```

```
data$Sepal.Length[132]
```

```
## [1] 7.9
```

```
data$Sepal.Length[which.max(data$Sepal.Length)]
```

```
## [1] 7.9
```

Slicing

Indexes in R start at 1 (as opposed to other languages, like Python, when they start at 0). Number of indexes needed to access a value/subset of values in the data structure in equal to the number of its dimensions:

- **vector**, **list** have 1 index; [**element_number**]
- **dataframe**, **matrix** have 2 indexes; [**row_number, column_number**]
- **nested list** have a number of indexes depending on a number lists nested

Let us get the first element of our vector.

```
vector
```

```
##  [1] 83 84 52  9 30 20 27 44 21 56 82 86 95 40 42  7 12 54 60 61
```

```
vector[1]
```

```
## [1] 83
```

Let us get the first element of our dataframe.

```
head(data)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
## 4          4.6         3.1          1.5         0.2  setosa
## 5          5.0         3.6          1.4         0.2  setosa
## 6          5.4         3.9          1.7         0.4  setosa
```

```r
data[1,1]
```

```
## [1] 5.1
```

Let us get the first element of an example of a nester list.

```r
list_1 <- list(1, 2, 45, 6)
list_2 <- list(LETTERS)
nested_list <- list(list_1, list_2)
nested_list[1]
```

```
## [[1]]
## [[1]][[1]]
## [1] 1
##
## [[1]][[2]]
## [1] 2
##
## [[1]][[3]]
## [1] 45
##
## [[1]][[4]]
## [1] 6
```

```r
nested_list[[1]]
```

```
## [[1]]
## [1] 1
##
## [[2]]
## [1] 2
##
## [[3]]
## [1] 45
##
## [[4]]
## [1] 6
```

```r
nested_list[[1]][[1]]
```

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

```r
nested_list[[2]]
```

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

LETTERS is a list itself!

```r
nested_list[[2]][[1]][1]
```

```
## [1] "A"
```

```r
unlist(nested_list)
```

```
##  [1] "1"  "2"  "45" "6"  "A"  "B"  "C"  "D"  "E"  "F"  "G"  "H"  "I"  "J"  "K"
## [16] "L"  "M"  "N"  "O"  "P"  "Q"  "R"  "S"  "T"  "U"  "V"  "W"  "X"  "Y"  "Z"
```

If nested lists are confusing, do not worry, we are not going to use them much! But it is good to know that they exist!

Subsetting options: a vector

```
vector
```

```
##  [1] 83 84 52  9 30 20 27 44 21 56 82 86 95 40 42  7 12 54 60 61
```

```
vector[c(1,3)]
```

```
## [1] 83 52
```

```
vector[-c(1,3)]
```

```
##  [1] 84  9 30 20 27 44 21 56 82 86 95 40 42  7 12 54 60 61
```

```
vector[c(2:5)]
```

```
## [1] 84 52  9 30
```

```
vector[2:5]
```

```
## [1] 84 52  9 30
```

All numbers except one variable.

```
vector[-3]
```

```
##  [1] 83 84  9 30 20 27 44 21 56 82 86 95 40 42  7 12 54 60 61
```

All numbers but the last four.

```
vector[-c(17:20)]
```

```
##  [1] 83 84 52  9 30 20 27 44 21 56 82 86 95 40 42  7
```

Boolean vector.

```
boolean <- sample(c(TRUE, FALSE), 20, replace=TRUE)
boolean
```

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

```
vector[boolean]
```

```
##  [1] 83 84 52 30 20 21 82 86 95 42 61
```

Subsetting options: a dataframe

```
head(data)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
## 4          4.6         3.1          1.5         0.2  setosa
## 5          5.0         3.6          1.4         0.2  setosa
## 6          5.4         3.9          1.7         0.4  setosa
```

Row 1 and 3, column 1 and 3.

```
data[c(1,3),c(1,3)]
```

```
##   Sepal.Length Petal.Length
## 1          5.1          1.4
## 3          4.7          1.3
```

All data except row 1 and 3, column 1 and 3.

```
data[-c(1,3), -c(1,3)]
```

First 2 rows, all columns.

```
data[c(1:2),]
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
```

First 2 rows, all columns.

```
data[1:2,]
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
```

All rows except the first one, all columns.

```
data[-1, ]
```

All columns except the last one, all rows.

```
data[, -1]
```

Boolean vectors.

```
boolean_rows <- sample(c(TRUE, FALSE), 150, replace=TRUE)
boolean_columns <- sample(c(TRUE, FALSE), 5, replace=TRUE)
data[boolean_rows, boolean_columns]
```

Variable names.

```
data[1:3, 'Species']
```

```
## [1] setosa setosa setosa
## Levels: setosa versicolor virginica
```

```
data$Species[1:3]
```

```
## [1] setosa setosa setosa
## Levels: setosa versicolor virginica
```

Finally, you can use **subset()** function.

```
subset(data, Species=='virginica')
```

Exercises.

1. What flower is the 55th observation in the iris dataframe?
2. Print out all values of **Petal.Length** columns of the dataframe.

3. Get a subset of a dataframe with 3 rows and all 5 columns. Call it ‘my_subset’. What flowers you got there? Pick the flower with greatest Sepal.Length. What is the index of that species’ name in your subset?

Congrats!

You are ready to slice dataframes and vectors!