Machine Learning - 1100-ML0ENG (Ćwiczenia informatyczne Z-23/24)

Introduction to R

Demo

```
demo(graphics)
demo(persp)
```

Available objects and their removal

Command **ls()** or **objects()** list the objects in the current workspace

```
ls()
objects()
```

Removing objects using **rm ()**

```
rm(z)
rm(list = ls()) #all objects
```

Document outline

R makes use of the **# sign to add comments,** so that you and others can understand what the R code is about.

Code Sections

Code sections allow you to break a larger source file into a set of discrete regions for easy navigation between them.

To insert a new code section you can use the **Code -> Insert Section** command. Alternatively, any comment line which includes at least four trailing dashes (-), equal signs (=), or pound signs (#) automatically creates a code section. For example, all of the following lines create code sections:

```
# Section One ------------------------------

# Section Two ==============================

### Section Three ############################
```

Note that as illustrated above the line can start with any number of pound signs (#) so long as it ends with four or more -, =, or # characters.

Help in R

Let us find some information about command solve

```
help(solve)
#or
?solve
```

R as a calculator

```
##################################################
### Simple manipulations numbers
##################################################
```

```
2+2
2^10 - 1 #exponentiation
5 %% 3 #modulo returns the remainder of the division of the number to the left by
the number on its right
log(1024, 2)
sin(pi/3)^2 + cos(pi/3)^2
```

```
exp(1)
```

```
log(exp(3))
pi
sqrt(4)
floor(6.7)
```

```
ceiling(6.7)
```

```
round(6.76575, digits = 1)
```

Variable assignment

A variable allows you to store a value or an object in R. You can then later use this variable's name to easily access the value or the object that is stored within this variable. You can assign a value 5 to a variable var_1 with the command

```
# "=" & "<-" assignment operators
(x = round(6.76575, digits = 1))
```

```
var_1 <- 5
var_1 # print out the value of the variable var_1
```

We can create variables with **dot** in the name

```
var.1 <- 5
var.1 # print out the value of the variable var.1
```

```
a <- 3
b <- 5
a + b
c <- a/b + 2*b + 1 # assign the result of operations to a new variable c
```

Assignment operators can be freely joined.

```
var1=var2 <- 2+2 -> var3
```

all variables contained 4.

Creating variables of different Types

R works with numerous data types. Some of the most basic types to get started are:

- Decimals values like 4.5 are called numeric.
- Natural numbers like 4 are called integers. Integers are also numeric.
- Boolean values (TRUE or FALSE) are called logical.
- Text (or string) values are called characters.

```
#####################################################
### Data Types in R
#####################################################
# Double (is numeric)
# Integer (is numeric)
# Complex
# Logical
# Character
# Factor
# Dates and Times
```

```
is.double(as.integer(67))
```

```
# Double Double precision scalar, vector
 is.double(as.integer(4.5)/as.integer(2))
```

```
my_numeric <- 42
my_character <- "forty-two" #we use quotation marks
my_logical <- FALSE
```

```
# Complex - Complex scalars or Vectors
```

```
is.complex(5)
```

```
#Different way of defining complex number
```

```
complex(real = 7,imaginary = 6)
```

```

```

Error

```
# Assign a value to the variable my_apples
my_apples <- 5

# Fix the assignment of my_oranges
my_oranges <- "six"

# Create the variable my_fruit and print it out
my_fruit <- my_apples + my_oranges
my_fruit
```

You would be trying to assign the addition of a numeric and a character variable to the variable my_fruit. **This is not possible.**

Infinite and NaN Numbers

NaN means Not a Number.

```
a<-log(-3)
a
```

Inf and -Inf are positive and negative infinity.

```
1/0
-2/0
# Special symbols: NA, TRUE, FALSE, NULL, NaN
```

Vectors

Vectors are one-dimension arrays that can hold numeric data, character data, or logical data. In other words, a vector is a simple tool to store data.

In R, you create a vector with the combine **function c()**. You place the vector elements separated by a comma between the items.

```
numeric_vector <- c(1, 2, 3)
character_vector <- c("a", "b", "c")
boolean_vector <- c(TRUE, FALSE, TRUE)
```

Once you have created these vectors in R, you can use them to do calculations.

```
sum(numeric_vector)
```

Arithmetic Operators

Operator	Description
+	addition
-	subtraction
*	multiplication
/	division
^	exponentiation
x %% y	modulus (x mod y) 5%%2 is 1
x %/% y	integer division 5%/%2 is 2

Practise

Consider two vectors: x, y

```
x<-c(4,6,5,7,10,9,4,15)
y<-c(0,10,1,8,2,3,4,1)
```

compute:

```
x+y, x-y, x*y, x/y, x^y, x%%y, x%/%y
```

```
# generating regular sequences
```

```
# Sequence Generation
```

```
seq(1, 10, by = 0.5)
```

```
seq(1, 10, length.out = 6)
```

```
1:20
# Replicate Elements of Vectors and Lists
rep(6, 4)
rep(1:5, each = 4, times = 5)
rep(1:5, 4, each = 5)
```

Vector indexes

Individual elements of a vector are accessed using the index **operator [ ].**

```
x = (1:5) ^ 2
x[1]
x[2:3]
x[c(2,4)]
x[c(2, 4, 4)]
```

A vector with negative indices indicates values to be excluded.

```
x[-1]
```

```
x[-(2:3)]
```

```
x[c(-1,2)]#error
```

```
x[0]
```

```
x[-(1:4)]<- 100
```

The elements of this vector do not have any name, but you can give them a name. If the elements of a vector have a name, then it is called a named vector. For example, you can give the name of the elements of the vector games as follows:

```
games2 <- c(g1="volleyball", g2="football", g3="basketball", favorite="handball")
games2["g1"]
games2["favorite"]
```

or

```
p<-3:8
p
names(p)<- c("first", "second", "third", "fourth", "fifth")
p
```

All types of vectors can be named. In this case a sub-vector of the names vector may be used in the same way as the positive integer labels
Interestingly, an index can also be a logical expression, in which case you get the elements for which the expression is TRUE.

```
x<-c(22,-5,5,90,-3,-4,4,17)
x[x>0]
x[x %% 2 == 0 & x>4]
z<-x[(x<-4) | (x>=22)]
```

The **which()** function returns the locations where a logical vector is TRUE. This can be useful for switching from logical indexing to integer indexing:

```
which(x > 10)
```

which.min and which.max are more efficient shortcuts for `which(min(x))` and `which(max(x))`, respectively:

```
which.min(x)
which.max(x)
```

```
# Changing the length of an object
```

```
y=9
length(9)
y[8]=0
```

```
length(y)=3
y
```

Last modified: środa, 4 października 2023, 7:51

Accessibility settings

Przetwarzanie danych osobowych

Platformą administruje Komisja ds. Doskonalenia Dydaktyki wraz z Centrum Informatyki Uniwersytetu Łódzkiego Więcej

Informacje na temat logowania

Na platformie jest wykorzystywana metoda logowania za pośrednictwem Centralnego Systemu Logowania.

Studentów i pracowników Uniwersytetu Łódzkiego obowiązuje nazwa użytkownika i hasło wykorzystywane podczas logowania się do systemu USOSweb.

Deklaracja dostępności