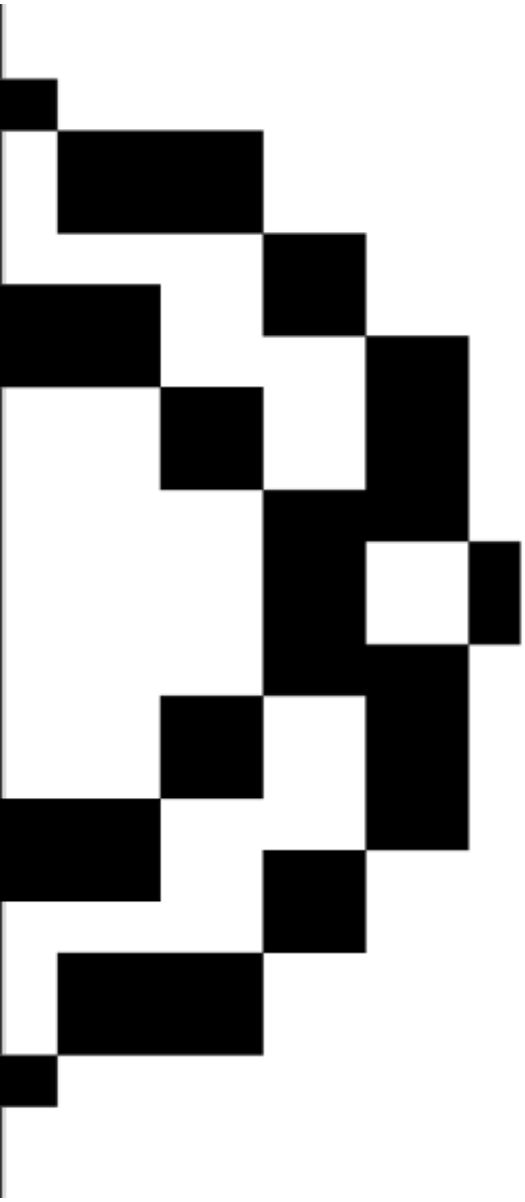
NOVA IMS INFORMATION MANAGEMENT SCHOOL



Exercises: Introduction to R

Statistics for Data Science

Master Program in Advanced Analytics

2025/2026

1 Objects in R

1.1 Expressions

An expression in **R** corresponds to a sequence of operations, and it can include variables and/or functions. These expressions can be evaluated in **R** and have (only one) return.

1. Consider the following expressions:

   - `123 * 321`
   - `42 ** 2`
   - `34 - 2 * 6`
   - `4 + 2 == 42`
   - `1 + sum(2,3)`

   (a) Assess the result of these expressions in **R**.

   (b) Save the result of the last expression in a variable named `x` (Important: R is case sensitive! Pay attention whether you are using upper or lower case).

   (c) Use that same variable, `x`, in another expression.

   (d) Attribute to the variable `x` the value 42. What happened to the result of the expression in (b)?

1.2 Sequences

1. Check the function `seq` in help and create the following sequences:

   - from 1 to 20
   - from 1 to 20, by 2
   - 15 elements from 1 to 20
   - 20 elements from 1, spaced by 0.4
   - number 1 repeated 100 times

1.3 Vectors

Vectors are a simple yet powerful way to store information in **R**. However, these objects can only store one type of information (i.e, numeric, logical, character). Vectors are created through the function `c()`.

1. Consider the following vectors:
   `a = c("L", 3, 3, T)` and `b = c(F, 4, 1, 7)`
   Create these vectors in R and explain what happened to values saved in both vectors.

2. Consider the following Table:

Table 1: Grades

Name	**Grade**
Andre	10
Carolina	12
John	-
Pedro	15
Zé	15
Vânia	7

   (a) Save the information in the Table 1 in two separate vectors: `names` and `grades`. Beware of the missing grade. What would happen if you saved the data in the same vector?

   (b) Verify, with a logical operator, if the two vectors have the same size (Hint: Check the function `length`).

   (c) To access an element of a vector, we use the notation: `VectorName[i]`, where i is the index of element you want to access. In R, the index count starts with 1 (not with zero!).
   Knowing this, access the name and grade of the 5^{th} student.

   (d) Update the missing grade with 13.

1.4 Matrices

A matrix, in **R**, is an extension of a vector to two dimensions.

1. Consider the following vector that contains the grades of a curricular unit:
   `grades = sample(8:20, 20)`
   Verify its dimensions, using the function `dim()`. Transform it into a matrix `grades`, with 4 lines and 5 columns. Now, verify the dimensions of the matrix created.

2. Add labels to the columns and lines of the matrix `grades`, using the function `colnames()` and `rownames()`.

3. To acess elements within a matrix, we use the notation: `MatrixName[i,j]`, where i represents the position/index or name of the row and j is the position/index or name of column.
   Access the elements of the matrix `grades` in different ways:

   - by two positions
   - by two names
   - by one position and one name
   - a whole line by a position
   - a whole column by a name
   - the first three lines and all the columns except the last one.

4. Assess the result of some operations on the following matrix:

   - grades + 1
   - grades * 2
   - grades * grades
   - grades %*% grades
   - determinant, through the function det()
   - transpose, through the function t()
   - invert, through the function solve()

1.5 Data Frames

The structure of data `data.frame` is one of the most used in **R**, mainly due to its versatility in the type of objects supported and in indexation: it is derived from lists, supporting elements from different types in a single object; and it adds the functionalities of indexation of vectors.
It is mainly used to keep tables of data, being similar to a matrix with names in lines and columns, but it supports elements from different types. This type of table can be seen as a database, or a spreadsheet, where each line corresponds to a different register.

1. Consider the table below:

Name	Favorite_Tech	Years_Experience
Ana	R	3
Bruno	Python	5
Clara	Power BI	2
Daniel	Tableau	4

   Transform it into a data frame using the function `data.frame()`.

2. Check the dimensions of the data frame created.
3. Show the lines where individuals have more than 3 years of experience.
4. Update Clara's favorite tech to Python.

2 Control Structures

Control structures allow you to control the flow of execution of a script typically inside of a function. Some common structures include: `if`, `else`, `for`, `while` and `repeat`, `break`. We do not use these while working with R interactively but rather inside functions.

1. Without using **R**, state the expected results given by the following chunks of code:

   (a)
   ```
   ifelse(sqrt(9)<2,sqrt(9),0)
   ```

   (b)
   ```
   ifelse(sqrt(100)>9,sqrt(100),0)
   ```

   (c)
   ```
   x=12
   if(is.numeric(x)) y=x*2
   ```

   (d)
   ```
   z=-1
   if(z<0){x=abs(z);y=z*3}
   ```

   (e)
   ```
   z=6
   if(z<0) y=z*3 else y=z*5
   ```

   (f)
   ```
   x=15
   y=3
   if(is.numeric(x))
   if(is.numeric(y) & y!=0)
   z=x/y
   ```

   (g)
   ```
   x=letters[20]
   if (is.numeric(x)) print('is numeric') else
   if(is.character(x)) print('is character')
   ```

   (h)
   ```
   x=90
   ifelse(x<100,ifelse(x/2==trunc(x/2),x/2,0),
   ifelse(x/100==trunc(x/100),x/100,-1))
   ```

2. Create a loop through `for` that prints all numbers from 1 to 15, except the number 3. Check in the examples available in help the notation of `for`.

3 Functions

Functions are used to logically break our code into simpler parts which become easy to maintain and understand. It is pretty straightforward to create your own function in R programming. The syntax for writing functions in R is:

```
  func_name <- function(argument) {
     statement
}
```

We have seen here several functions available within R, such as `sum()`. In R programming you can write and create your own functions to solve an existing problem or optimize your script. Consider this example to print x to the power of y:

```
pow <- function(x, y) {
   result <- x^y
   print(paste(x,"raised to the power", y, "is", result))
}
```

When creating a function, the best option is to start writing the statement, from which we identify the necessary arguments. So, in the previous example we write: $x\hat{y}$, from which we identify the arguments x and y. Then, we can print the result, just by typing the name (`result`) or use a a different format, such as with `paste()`.

1. Create the following functions:

   (a) A function called `propna` that receives a vector and returns the percentage of `NA` observations. Use the function `is.na()` to check the number of NA observations and calculate the proportion.

   (b) A function called `par` that receives a vector an excludes uneven numbers.

2. Create a function that converts Fahrenheit degrees (° F) to Celsius (° C), knowing that its relation is given by:

   $$^{\circ}C = (^{\circ}F - 32) \div 1.8$$

   Expand your function to convert Celsius degrees to Fahrenheit, optionally. The inverse relation is given by: $^{\circ}F = (^{\circ}C \times 1.8) + 32$