Practical Aspects of Database Design

L2 - R Session I

Stevens Institute of Technology

Object Types

Vector

When you want to create vector with more than one element, you should use c() function to combine the elements.

Matrix

A matrix is a two-dimensional rectangular data set. It can be created using a vector input to the matrix function.

Object Types

Array

- As array is made up matrices in multiple dimensions, the operations on elements of array are carried out by accessing elements of the matrices.
- An array is created using the array() function.
- It takes vectors as input and uses the values in the dim parameter to create an array.
- We can do calculations across the elements in an array using the apply() function

Object Types

Factor

- Used to categorize the data and store it as levels. They are useful in the columns which have a limited number of unique values. Like "Male, "Female".
- Factors are created using the factor () function by taking a vector as input. They can store both strings and integers.
- They are useful in data analysis for statistical modeling. Factors represent a very efficient way to store character values, because each unique character value is stored only once, and the data itself is stored as a vector of integers.

Object Types

Data Frame

- Data frame is useful to deal with data including different properties, such as personal data (name, age, address).
- If one column is factor in data frame, you can always use as.character() transferring it to char vector.
- Elements in data frame is vector (showing as column). You may have different data type among columns, but all values in one columns must have same type.
- You can set names on rows and columns in data frame (but not other variables).

Object Types

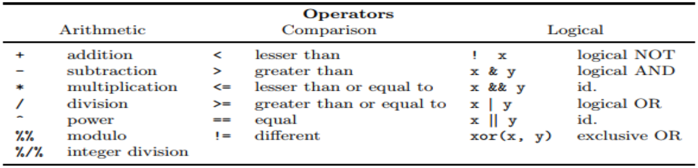
List

- Lists contain elements of different types like numbers, strings, vectors and another list inside it. A list can also contain a matrix or a function as its elements.
- List is created using list() function.

Object Types Transfers

- as.xxx() function can be used as coercion function to transfer the data type of a variable.
- Coercion function only give output of transferred result, but will not change the value of variable.
- NA means missing value. is.na() function can be used to check whether it is NA (a true or false will be returned).

Operations



Operators					
Arithmetic		Comparison		Logical	
`+`	addition	`<`	lesser than	`! x`	logical NOT
`-`	subtraction	`>`	greater than	`x & y`	logical AND
`*`	multiplication	`<=`	lesser than or equal to	`x && y`	id.
`/`	division	`>=`	greater than or equal to	`x | y`	logical OR
`^`	power	`==`	equal	`x || y`	id.
`%%`	modulo	`!=`	different	`xor(x, y)`	exclusive OR
`%/%`	integer division				

Functions

A function is created by using the keyword **function**. The basic syntax of an R function definition is as follows:

```
function_name <- function(arg_1, arg_2, ...) {
   Function body
}
```

Function components

- **Function Name**: This is the actual name of the function.
- **Arguments**: An argument is a placeholder. When a function is invoked, you pass a value to the argument. Arguments are optional;
- **Function Body**: contains a collection of statements that defines what the function does.
- **Return Value**: The return value of a function is the last expression in the function body to be evaluated.

Functions

Built-in functions

R has many in-built functions which can be directly called without defining them first. seq(), mean(), max(), sum() and paste()...

User-defined Function

We can create user-defined functions in R. They are specific to what a user wants and once created they can be used like the built-in functions.

- Calling a Function without Argument
- Calling a Function with Argument
- Calling a Function with Default Argument

Packages

- R packages are a collection of R functions, complied code and sample data.

Install a new package

- Install directly from CRAN(recommended)

```
install.package("package name")
```

- Install package manually
  - Go to the link R Packages to download the package needed. Save the package as a .zip file in a suitable location in the local system.
  - Run the following command to install this package.

```
install.packages(file name, repos = NULL, type = "source")
```

Packages

Load package to library

```
library("package Name", lib.loc = "path to library")
```

Available CRAN packages

https://cran.r-project.org/web/packages/available_packages_by_name.html

Extension: Making Your First R Package

`https://support.rstudio.com/hc/en-us/articles/200526207-Using-Projects`