R Programming

JOHN HOPKINS UNIVERSITY

**WEEK 1**

* The ‘<-‘ is used to assign a value as a vector to a variable. The ([1]) symbol before the value shows the index of the value in the vector.

|  |  |
| --- | --- |
|  |  |
| Numeric Vector | Character Vector |

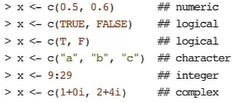
* The value of the variable can be checked in two different ways-

|  |  |
| --- | --- |
|  |  |
| Running Variable  (Auto-Printing) | Running print()  (Explicit-Printing) |

* The “#” is used to type in a comment.
* If a variable assignment line is run without inputting the value, then the script will wait for value assignment before proceeding.
* The “:” is used to create a sequence.



* The R language is made up of 5 classes of objects –
  1. Character
  2. Numeric (Real Numbers)
  3. Integers
  4. Complex
  5. Logical (True/False)
* Vector is the most basic type of object. It can only hold values of the same class.
* An only exception is the *List* type of vector. This is able to hold values of different classes.
* The *vector()* function can be used to create an empty vector.
* “L” is used to assign a numeric value to a variable instead of a numeric value in a vector.
* *“Inf”* represents infinity value.
* *‘NaN’* represents null value.
* All R objects have attributes-
  1. Names, Dimnames
  2. Dimensions (Eg. Matrices)
  3. Class
  4. Length
  5. Other user-defined metdata
* *“attributes()”* function allows accessing the attributes of an object.
* The *“c”* function is used to create vector objects.



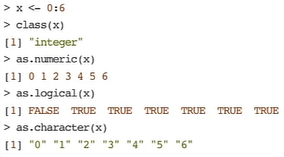
* The *“vector()”* function can also be used to create a vector object.



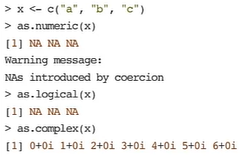
* Since vector objects can only hold values of the same class, if values input into the vector is of different class, then all the values are converted into the least denominator class.



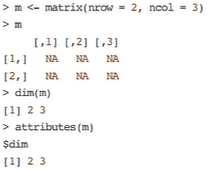
* The *“class()”* function give the class of the variable as the output.
* The *“as.\*”* function is used to explicitly convert the type of the value.



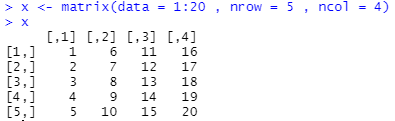
* Sometimes, the conversion does not work. Converting character values to integer does not work. This is called **Nonsensical Coercion (NAs)**.



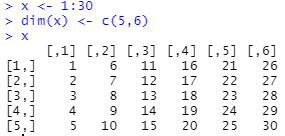
* The *“list()”* function can be used to create a list object.
* **Matrices** are another type of vector which has a **dimension** attribute. This attribute is an integer vector of length 2 (nrow , ncol).



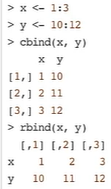
* Matrices are constructed column-wise.



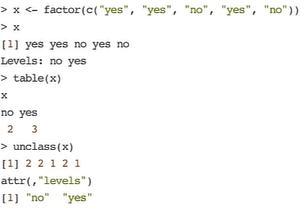
* Matrix vectors can be created by adding the dimension values after entering the values.



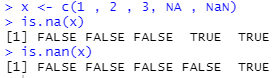
* Matrix vectors can also be created by assigning values to rows or columns using *“rbind()”* or *“cbind()”*.



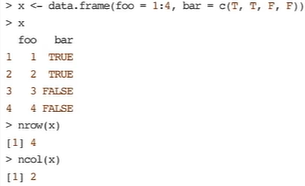
* **Factors** is another type of vector used to represent categorical data. It is of two types-
  + Ordered
  + Unordered
* The *“factor()* function is used to create the function vector.
* The *“table()”* function is used to find the numerical value of each value in factor.
* The *“unclass()”* function is used to convert the function values to numeric values.



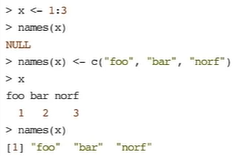
* The order of the levels can be set by using the *“levels”* arguement in *factor()* function.
* **Levels** are important because modelling functions takes the first value in **Levels** as the baseline.
* The order of values in **Levels** is usually in alphabetical order.
* Missing values are denoted using **“NA”** or **“NaN”**. **NaN** usually represents undefined mathematical operations.
* *“is.na()”* and *“is.nan()”* functions are used to check if the value is NA or NaN respectively.
* **Na** values can have different classes. Ex. Integer Na, Character Na etc.
* An **NaN** value is also an **NA** value but the converse is not true.



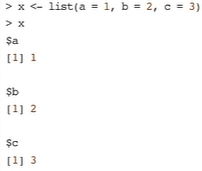
* **Data Frames** is another data type. It is used to store tabular data. They are represented as a special type of list with each element in the list having same length. Each list element is considered as a column and the length of each element is its rows.
* Data frames have a special attribute known as row.names(). Can be used to give names to the rows.
* Data Frames can also be created by calling files using read.table() or read.csv().
* Data Frames can be converted into matrix using data.matrix().
* Data Frames are usually created by using the *“data.frame()”* function.



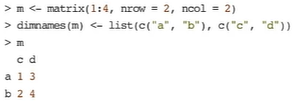
* **Names** can be given to the values in the vector associated to a variable.



* It is possible to name values in the list also.



* Even matrices can be named.



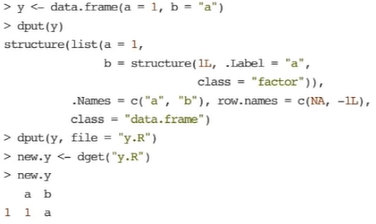
**READING DATA FILES USING read.table**

Arguments of the read.table() function –

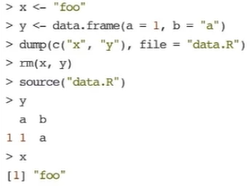
|  |  |
| --- | --- |
| *file* | Name of file |
| *header* | Logical indicating if file has header |
| *sep* | String indicating how columns are separated. |
| *colClasses* | Character vector showing the class of each column |
| *nrows* | No. of rows in dataset |
| *comment.char* | Character string indicating the comment character. |
| *skip* | No. of line to skip from the beginning. |
| *stringasFactors* | Should character variables be coded as factors. |

**TEXTUAL DATA FORMATS**

The *“dput()”* function creates an R code for recreating an R object.



The *“dump()”* function is similar to dput(), except it can create R code to reconstruct multiple R objects. The *“source()”* function is used to retrieve and reconstruct the objects using R code.

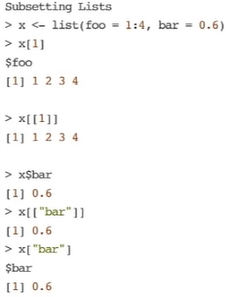


**SUBSETTING**

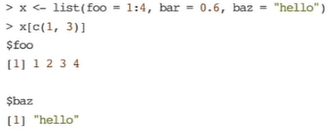
**[ ]** : Used to subset the same type of class as the set.

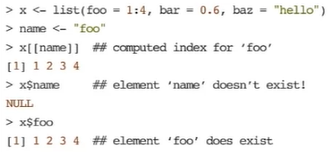
**[[ ]]** : Used to subset in lists.

**$** : Used to subset in lists using names. It is different from [[ ]] in that $ requires the usage of names, whereas [[ ]] reuires the index.

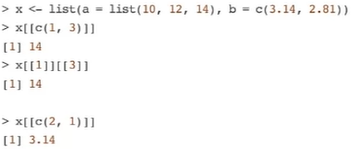


Multiple items in a list can be subset using c().

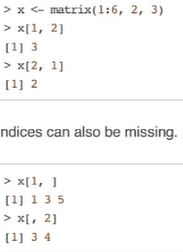




Subsetting nested elements can be done by –

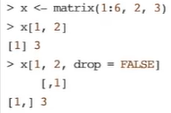


Subsetting a matrix can be done like this –

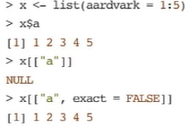


Matrices are interesting in that the **[ ]**  does not always return a matrix like it should. Instead, if a 1x1 matrix is subset, the value is returned instead of a matrix. This is the same when subsetting a single row or column.

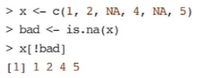
This can be solved by using the *drop* argument.



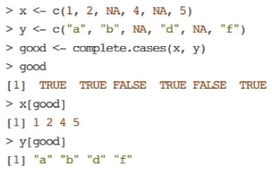
Partial matching of names can be done by both **[[ ]]** & **$**.



A common method to remove NA values is –



Another method to remove NA values is by using the *“complete.cases()”* function.



This function can also be used to remove NA values in matrices.

