R Notebook

http://rmarkdown.rstudio.com/r_notebooks.html

This is an R Markdown Notebook. When you execute code within the notebook, the results appear beneath the code.

Try executing this chunk by clicking the *Run* button within the chunk or by placing your cursor inside it and pressing *Ctrl+Shift+Enter*.

```
d1 <- 2+3
d1
```

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

```
array_1 <-array(1:24, dim=c(3,4,2))
array_1
```

```
## , , 1
##
##      [,1] [,2] [,3] [,4]
## [1,]    1    4    7   10
## [2,]    2    5    8   11
## [3,]    3    6    9   12
##
## , , 2
##
##      [,1] [,2] [,3] [,4]
## [1,]   13   16   19   22
## [2,]   14   17   20   23
## [3,]   15   18   21   24
```

Creating our very First Array

```
?array()

array_1 <-array(1:24, dim=c(3,4,3))
array_1
```

```
## , , 1
##
##      [,1] [,2] [,3] [,4]
## [1,]    1    4    7   10
## [2,]    2    5    8   11
## [3,]    3    6    9   12
##
## , , 2
##
##      [,1] [,2] [,3] [,4]
## [1,]   13   16   19   22
## [2,]   14   17   20   23
## [3,]   15   18   21   24
##
## , , 3
##
##      [,1] [,2] [,3] [,4]
```

```
## [1,]    1    4    7   10
## [2,]    2    5    8   11
## [3,]    3    6    9   12
```

DIM == "Integer vector of length one or more giving the maximal indices in each dimension." Above we have ROWS or INDEX == 3 , COLUMNS == 4 and DIMENSIONS == 3 VALUES or OBSERVATIONS are filled in COLUMNS FIRST - ROWS NEXT

```
is.array(array_1)
```

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

```
array_2 <-array(1:24, dim=c(3,4))
array_2
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1    4    7   10
## [2,]    2    5    8   11
## [3,]    3    6    9   12
```

```
typeof(array_2)
```

```
## [1] "integer"
```

```
typeof(array_1)
```

```
## [1] "integer"
```

Both ARRAYS Stored as INTEGERS in Memory

WHAT ARE - VECTORS — One Dimension Arrays which can hold - NUMERIC , CHARACTER or LOGICAL DATA
Vectors can be of THREE Types or MODES - which means the DATA TYPE they can hold.
#
A VECTOR is created using the - COMBINE Function()
#
Lets see an example of each type below -

```
num_vector <- c(22,22,33,33,44)
num_vector
```

```
## [1] 22 22 33 33 44
```

```
char_vector <- c("x","d","c","f")
char_vector
```

```
## [1] "x" "d" "c" "f"
```

```
logical_vector <- c(TRUE,FALSE,TRUE,FALSE,FALSE,FALSE)
logical_vector
```

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

SCALARS - One Element Vectors - useful for holding CONSTANT values

```
a1 <- "FINANCE"
b1 <- "MARKETING"
c1 <- "SALES"
d1 <- 3.1416
a1
```

```
## [1] "FINANCE"
```

```
b1
```

```
## [1] "MARKETING"
```

```
c1
```

```
## [1] "SALES"
```

```
d1
```

```
## [1] 3.1416
```

We refer ELEMENTS of a VECTOR by mentioning their INDEX - STARTING at 1

Take an example of the "num_vector" from above , we REFER the 1st ELEMENT - Element 1 as below , assign it to a VAR "nm_1" and then print the value stored within the VAR :-

```
nm_1 <- num_vector[1]
nm_1
```

```
## [1] 22
```

Below we refer value stored in Elements - 2,3 and 5 , the values as seen below are - 22, 33 and 44

```
nm_2 <- num_vector[2]
nm_3 <- num_vector[3]
nm_5 <- num_vector[5]

nm_2
```

```
## [1] 22
```

```
nm_3
```

```
## [1] 33
```

```
nm_5
```

```
## [1] 44
```

We can also access a RANGE of elements by refering the INDEX of START and STOP Elements :-

```
nm_range <- num_vector[2:5]
nm_range
```

```
## [1] 22 33 33 44
```

MATRICES (Plural) MATRIX (Singular) - again like a VECTOR can store values in elements of the SAME MODE or SAME TYPE .

Matrices are TWO DIMENSIONAL ARRAYS of Data .

```
?matrix()
```

matrix(data = NA, nrow = 1, ncol = 1, byrow = FALSE, dimnames = NULL)

```
m_1 <- matrix(data=66:99,nrow=2,ncol=2)
m_1
```

```
##      [,1] [,2]
## [1,]   66   68
## [2,]   67   69
```

Data set has been truncated as we didnt mention enough ROWS for the MATRIX , lets try again with 5 ROWS ....

```
m_1 <- matrix(data=66:99,nrow=5,ncol=2)
```

```
## Warning in matrix(data = 66:99, nrow = 5, ncol = 2): data length [34] is
## not a sub-multiple or multiple of the number of rows [5]
```

```
m_1
```

```
##      [,1] [,2]
## [1,]   66   71
## [2,]   67   72
## [3,]   68   73
## [4,]   69   74
## [5,]   70   75
```

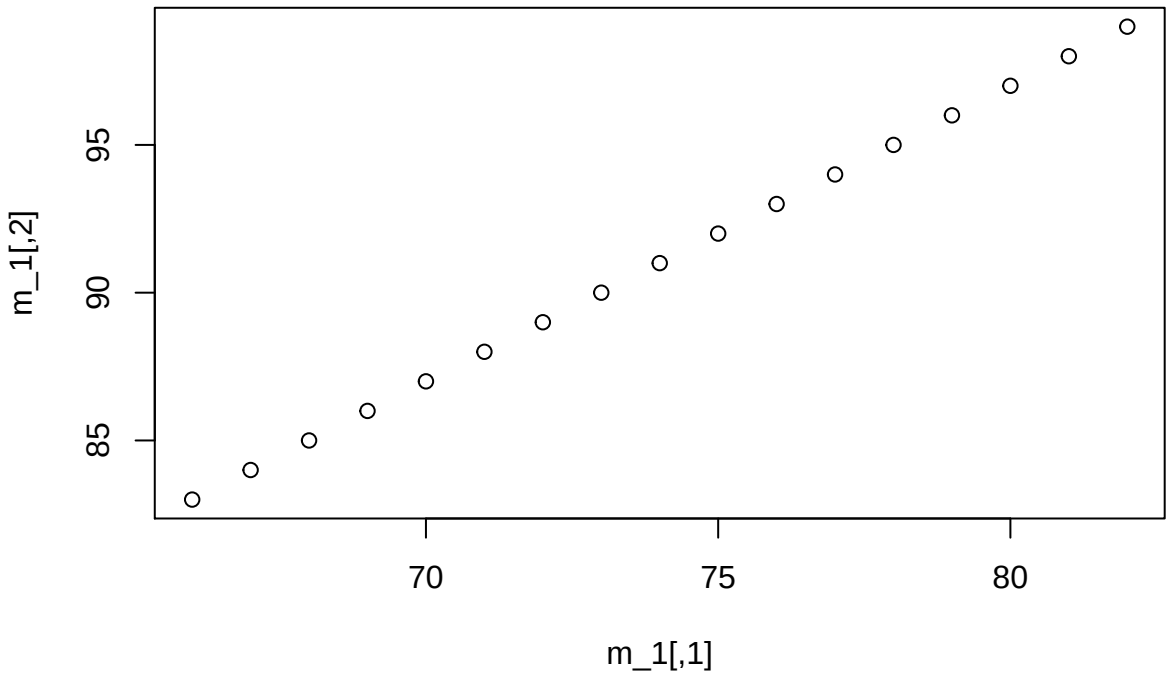
For Data Length 34 (data being numeric values from 66 to 99) , we need to provide a "multiple" or "sub-multiple"of 34 as the ROWS value - lets try with 17.

```
m_1 <- matrix(data=66:99,nrow=17,ncol=2)
m_1
```

```
##       [,1] [,2]
##  [1,]   66   83
##  [2,]   67   84
##  [3,]   68   85
##  [4,]   69   86
##  [5,]   70   87
##  [6,]   71   88
##  [7,]   72   89
##  [8,]   73   90
##  [9,]   74   91
## [10,]   75   92
## [11,]   76   93
## [12,]   77   94
## [13,]   78   95
## [14,]   79   96
## [15,]   80   97
## [16,]   81   98
## [17,]   82   99
```

As seen above now the MATRIX - m_1 , can fit in the Data Values just fine with - 17 ROWS and 2 COLUMNS.

```
plot(m_1)
```



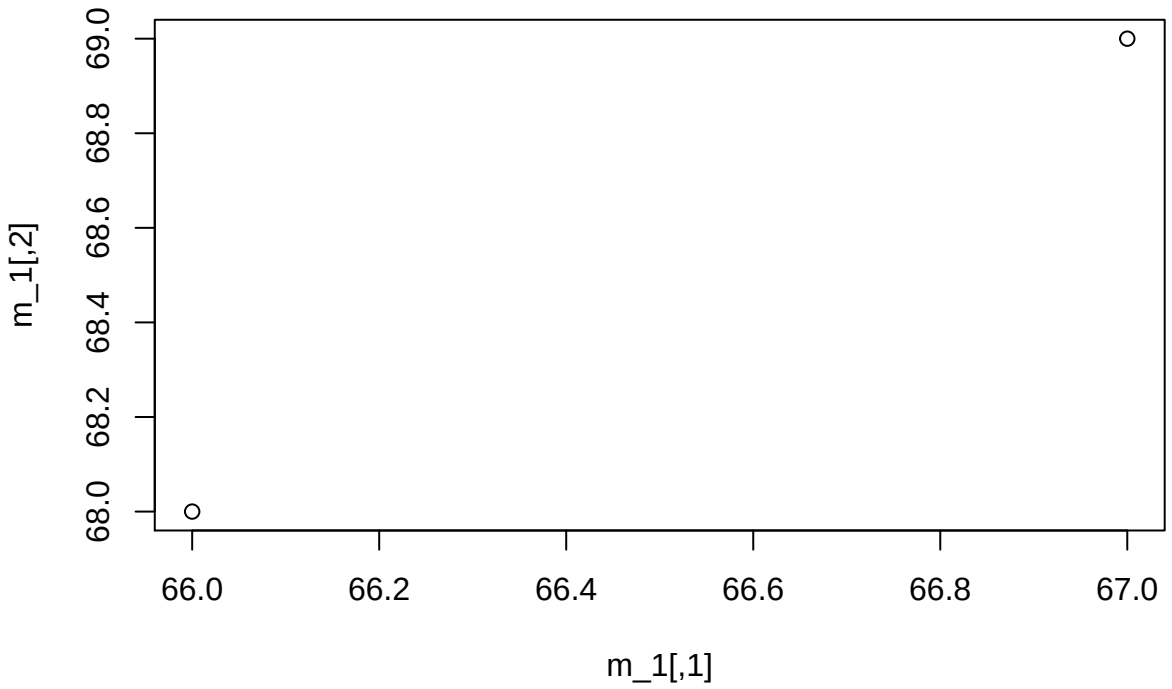
We can also reverse the ORDER of COLUMNS and ROWS

```
m_1 <- matrix(data=66:99,nrow=2,ncol=17)
m_1
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13]
## [1,]   66   68   70   72   74   76   78   80   82    84    86    88    90
## [2,]   67   69   71   73   75   77   79   81   83    85    87    89    91
##      [,14] [,15] [,16] [,17]
## [1,]    92    94    96    98
## [2,]    93    95    97    99
```

We still derive a MATRIX - m_1 , but the plot as seen below is illegible.

```
plot(m_1)
```



Weve seen the basic arguments for the Function “matrix()” above - now we look at “byrow” and “dimnames”. DimNames or Dimension Names can be a LIST of structure - LIST(ROW_NAMES,COLUMN_NAMES) . If we have only the 1st ELEMENT of a LIST then that defaults to ROW_NAMES and if we have a 2nd ELEMENT it defaults to COLUMN_NAMES.
Seen below - row_names and col_names created using the COMBINE - c() , function.

```
row_names <- c("Row_Name_1","Row_Name_2")
col_names <- c("Col_Name_1","Col_Name_2")
m_1 <- matrix(data=66:69,nrow=2,ncol=2,byrow=FALSE,dimnames=list(col_names,row_names))
m_1
```

```
##            Row_Name_1 Row_Name_2
## Col_Name_1         66         68
## Col_Name_2         67         69
```

sessionInfo()

R version 3.3.2 (2016-10-31) Platform: x86_64-pc-linux-gnu (64-bit) Running under: Ubuntu 16.04.1 LTS

locale: [1] LC_CTYPE=en_IN.UTF-8 LC_NUMERIC=C LC_TIME=en_IN.UTF-8 LC_COLLATE=en_IN.UTF-8
[5] LC_MONETARY=en_IN.UTF-8 LC_MESSAGES=en_IN.UTF-8 LC_PAPER=en_IN.UTF-8 LC_NAME=C
[9] LC_ADDRESS=C LC_TELEPHONE=C LC_MEASUREMENT=en_IN.UTF-8 LC_IDENTIFICATION=C

attached base packages: [1] stats graphics grDevices utils datasets methods base

loaded via a namespace (and not attached): [1] backports_1.0.4 magrittr_1.5 rprojroot_1.1 htmltools_0.3.5 tools_3.3.2 base64enc_0.1-3 yaml_2.1.14
[8] Rcpp_0.12.8 stringi_1.1.2 rmarkdown_1.3 knitr_1.15.1 jsonlite_1.1 stringr_1.1.0 digest_0.6.10
[15] evaluate_0.10

EOF - R_1.Rmd