QUEEN'S UNIVERSITY BELFAST
SCHOOL OF ELECTRONICS, ELECTRICAL ENGNIEERING AND COMPUTER SCIENCE

Data Analysis & Visualisation

CSC3062

BEng (CS & SE), MEng (CS & SE), BIT & CIT

Dr Reza Rafiee

Semester 1 – 2019/2020

This is R

R Programming; R-object

- In contrast to other programming languages like C and java, the variables in R are **not declared** as some data type. The **variables** are assigned with **R-Objects** and the data type of the R-object becomes the data type of the variable. There are many types of R-objects.

- The frequently used **R-Objects**:
  - Vectors
  - Lists
  - Matrices
  - Arrays
  - Factors
  - Dataframes

R Programming; Vectors

- The frequently used **R-Objects**:
  - Vectors
  - Lists
  - Matrices
  - Arrays
  - Factors
  - Dataframes

R Programming; Working directory & workspace

The working directory is the default place where R looks for files that are read from disk, or written to disk. The current working directory is obtained with:

```
> getwd()
[1] "C:/Users/1234567/Documents/Rwork"
```

We can also set the working directory using the function setwd()

```
> setwd("C:/Users/1234567/Documents")
```

Saving current session of R (workspace including all objects in memory) by save.image()

```
> save.image("myWspace1.RData")
```

Alternatively, you could use function load() to load your already saved workspace

```
> load("C:/Users/1234567/Documents/myWspace1.RData")
```

In the line below, we avoid R asking again whether it should save the workspace when using quit.

```
> q(save = "no")
```

R Programming; Vectors

Definition:

A string or numbers, sequential numbers, random numbers and so on

```
#-----------------------------------------------------------
# Some examples (running in the console)
#-----------------------------------------------------------
> VecNum1 <- vector(length=10, mode= "double")
[1] 0 0 0 0 0 0 0 0 0 0
#-----------------------------------------------------------
> VecLog <- vector(length= 5)
[1] FALSE FALSE FALSE FALSE FALSE
#-----------------------------------------------------------
> VecNum2 <- c(0,0,0,0,0,0,0,0,0,0)      # the easiest way to create a vector using c()
> VecNum2 <- c(rep(10,x=0))  # replicates elements of vectors and lists
[1] 0 0 0 0 0 0 0 0 0 0
#-----------------------------------------------------------
> SeqVec <- 1:100      # creates a sequence of numbers (from 1 to 100) – consecutive numbers
[1] 1 2 3 … 100
```

R Programming; Vectors & access to the elements

```
> x_vector <- seq(8,20,length.out=6)
> x_vector
[1] 8.0 10.4 12.8 15.2 17.6 20.0
#-------------------------------------------------------------
# Access to 3rd element of x_vector
> x_vector[3]
[1] 12.8
#-------------------------------------------------------------
# How to access to 2nd and 4th element of x_vector?

> x_vector[2,4]    # is it a correct call?
```

R Programming; Vectors & access to the elements

```
> x_vector <- seq(8,20,length.out=6)
> x_vector
[1] 8.0 10.4 12.8 15.2 17.6 20.0
#-------------------------------------------------------------
# Access to 3rd element of x_vector
> x_vector[3]
[1] 12.8
#-------------------------------------------------------------
# How to access to 2nd and 4th element of x_vector?

> x_vector[2,4]    # is it a correct call? No

Error in x_vector[2,4] : incorrect number of dimensions

> x_vector[c(2,4)]  # this is the correct one
#-------------------------------------------------------------
# How to access to all elements but 1st element?
```

R Programming; Vectors & access to the elements

```
> x_vector <- seq(8,20,length.out=6)
> x_vector
[1] 8.0 10.4 12.8 15.2 17.6 20.0
#-------------------------------------------------------------
# Access to 3rd element of x_vector
> x_vector[3]
[1] 12.8
#-------------------------------------------------------------
# How to access to 2nd and 4th element of x_vector?

> x_vector[2,4]    # is it a correct call? No

Error in x_vector[2,4] : incorrect number of dimensions

> x_vector[c(2,4)]  # this is the correct one
#-------------------------------------------------------------
# How to access to all elements but 1st element?
> x_vector[-1]    # [1] 10.4 12.8 15.2 17.6 20.0
```

R Programming; Vectors & access to the elements

```
> x_vector <- seq(8,20,length.out=6)
> x_vector
[1] 8.0 10.4 12.8 15.2 17.6 20.0
> typeof(x_vector)

> length(x_vector)

#-------------------------------------------------------------
> x_vector[c(2.1,4.5)]      # real numbers are truncated to integers

#-------------------------------------------------------------
> x_vector_char <- c(11, 50, TRUE, 'hello')     # what if we use this: c(11, 50, TRUE, "hello") ?

> typeof(x_vector_char)

#-------------------------------------------------------------
> x_vector <- seq(1,3,by=0.2)  # specify step size
```

R Programming; Vectors & access to the elements

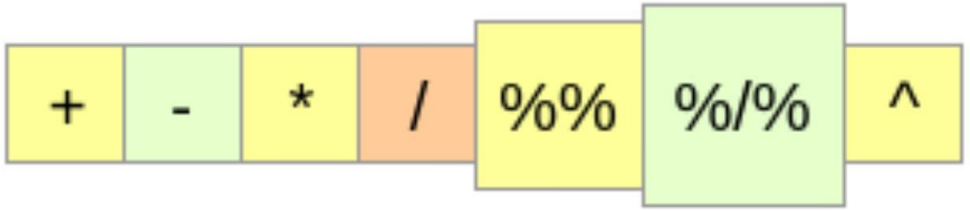
```
> x_vector <- seq(8,20,length.out=6)
> x_vector
[1] 8.0 10.4 12.8 15.2 17.6 20.0
> typeof(x_vector)
[1] "double"
> length(x_vector)
[1] 6
#-------------------------------------------------------------
> x_vector[c(2.1,4.5)]      # real numbers are truncated to integers
[1] 10.4 15.2
#-------------------------------------------------------------
> x_vector_char <- c(11, 50, TRUE, 'hello')     # what if we use this: c(11, 50, TRUE, "hello") ?
[1] "11"   "50"   "TRUE"   "hello"
> typeof(x_vector_char)
[1] "character"
#-------------------------------------------------------------
> x_vector <- seq(1,3,by=0.2)  # specify step size
[1] 1.0  1.2  1.4  1.6  1.8  2.0  2.2  2.4  2.6  2.8 3.0
```

R Programming; Vectors & access to the elements

```
> x_vec_logic <- c(TRUE,FALSE,TRUE,FALSE,FALSE)
> x_vec_logic
[1] TRUE   FALSE   TRUE   FALSE   FALSE
> typeof(x_vec_logic)
[1] "logical"
> length(x_vec_logic)
[1]  5
#-------------------------------------------------------------
```

R Programming; Operators

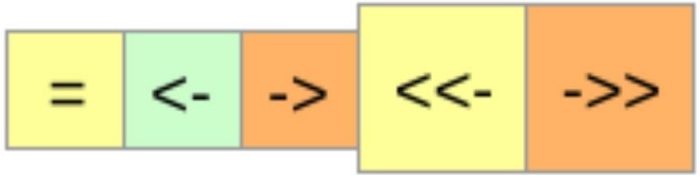
Arithmetic operators



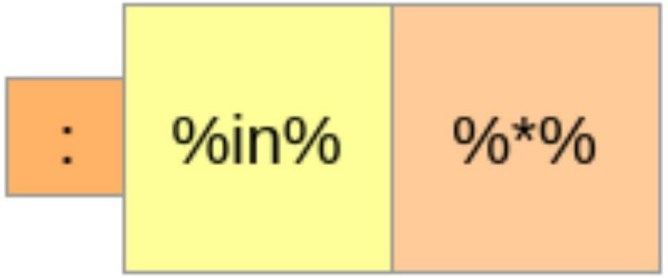
Relational operators

Logical operators

Assignment operators



Misc. operators



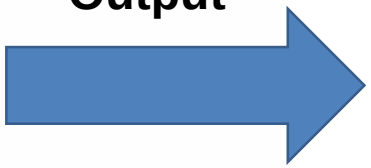
R Programming; Arithmetic operators

Operator	Description	Usage
+	Addition of two operands	a + b
–	Subtraction of second operand from first	a – b
*	Multiplication of two operands	a * b
/	Division of first operand with second	a / b
%%	Remainder from division of first operand with second	a %% b
%/%	Quotient from division of first operand with second	a %/% b
^	First operand raised to the power of second operand	a^b

R Programming; Arithmetic operators

```
# R Arithmetic Operators Example for integers

a <- 7.5
b <- 2

print ( a+b ) #addition
print ( a-b ) #subtraction
print ( a*b ) #multiplication
print ( a/b ) #Division
print ( a%%b ) #Reminder
print ( a%/%b ) #Quotient
print ( a^b ) #Power of
```

Output

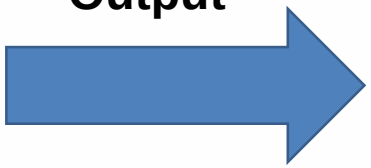


```
[1] 9.5
[1] 5.5
[1] 15
[1] 3.75
[1] 1.5
[1] 3
[1] 56.25
```

R Programming; Arithmetic operators

```
# R Operators - R Arithmetic Operators Example for vectors

a <- c(8, 9, 6)
b <- c(2, 4, 5)

print ( a+b ) #addition
print ( a-b ) #subtraction
print ( a*b ) #multiplication
print ( a/b ) #Division
print ( a%%b ) #Reminder
print ( a%/%b ) #Quotient
print ( a^b ) #Power of
```

Output



```
[1] 10 13 11
[1] 6 5 1
[1] 16 36 30
[1] 4.00 2.25 1.20
[1] 0 1 1
[1] 4 2 1
[1] 64 6561 7776
```

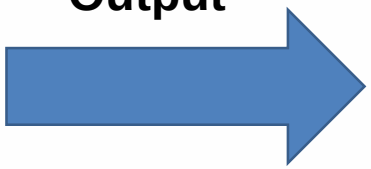
R Programming; Relational operators

Operator	Description	Usage
<	Is first operand **less than** second operand	a < b
>	Is first operand **greater than** second operand	a > b
==	Is first operand **equal to** second operand	a == b
<=	Is first operand **less than** or equal to second operand	a <= b
>=	Is first operand **greater than** or equal to second operand	a > = b
!=	Is first operand **not equal** to second operand	a!=b

R Programming; Relational operators

```
# R Operators - R Relational Operators Example for Numbers

a <- 7.5
b <- 2

print ( a<b ) # less than
print ( a>b ) # greater than
print ( a==b ) # equal to
print ( a<=b ) # less than or equal to
print ( a>=b ) # greater than or equal to
print ( a!=b ) # not equal to
```

Output

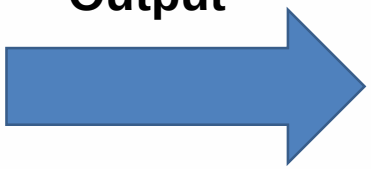


```
[1] FALSE
[1] TRUE
[1] FALSE
[1] FALSE
[1] TRUE
[1] TRUE
```

R Programming; Relational operators

```
# R Operators - R Relational Operators Example for Numbers

a <- c(7.5, 3, 5)
b <- c(2, 7, 0)

print ( a<b ) # less than
print ( a>b ) # greater than
print ( a==b ) # equal to
print ( a<=b ) # less than or equal to
print ( a>=b ) # greater than or equal to
print ( a!=b ) # not equal to
```

Output



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

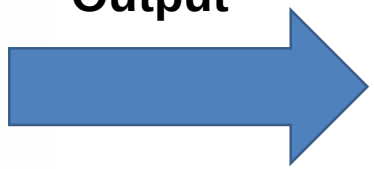
R Programming; Logical operators

Operator	Description	Usage
&	Element wise logical AND operation.	a & b
|	Element wise logical OR operation.	a | b
!	Element wise logical NOT operation.	!a
&&	Operand wise logical AND operation.	a && b
||	Operand wise logical OR operation.	a || b

R Programming; Logical operators

```
# R Operators - R Logical Operators Example for basic logical
elements

a <- 0 # logical FALSE
b <- 2 # logical TRUE

print ( a & b ) # logical AND element wise
print ( a | b ) # logical OR element wise
print ( !a ) # logical NOT element wise
print ( a && b ) # logical AND consolidated for all elements
print ( a || b ) # logical OR consolidated for all elements
```

Output

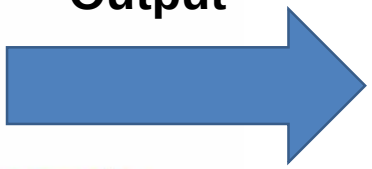


```
[1] FALSE
[1] TRUE
[1] TRUE
[1] FALSE
[1] TRUE
```

R Programming; Logical operators

```
# R Operators - R Logical Operators Example for boolean vectors

a <- c(TRUE, TRUE, FALSE, FALSE)
b <- c(TRUE, FALSE, TRUE, FALSE)

print ( a & b ) # logical AND element wise
print ( a | b ) # logical OR element wise
print ( !a ) # logical NOT element wise
print ( a && b ) # logical AND consolidated for all elements
print ( a || b ) # logical OR consolidated for all elements
```

Output



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

R Programming; Logical operators

& vs. &&

```
> ((-2:2) >= 0) & ((-2:2) <= 0)
[1] FALSE  FALSE  TRUE  FALSE  FALSE
```

-2 -1 0 1 2

```
> ((-2:2) >= 0) && ((-2:2) <= 0)
[1] FALSE
```

& is vectorised

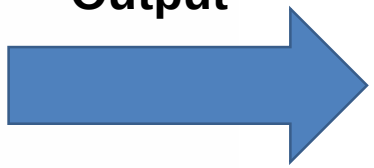
R Programming; Assignment operators

Operator	Description	Usage
=	Assigns right side value to left side operand	a = 3
<-	Assigns right side value to left side operand	a <- 5
->	Assigns left side value to right side operand	4 -> a
<<-	Assigns right side value to left side operand	a <<- 3.4
->>	Assigns left side value to right side operand	c(1,2) ->> a

R Programming; Assignment operators

```
# R Operators - R Assignment Operators

a = 2
print ( a )

a <- TRUE
print ( a )

454 -> a
print ( a )

a <<- 2.9
print ( a )

c(6, 8, 9) -> a
print ( a )
```

Output



```
[1] 2
[1] TRUE
[1] 454
[1] 2.9
[1] 6 8 9
```

R Programming; Misc. operators

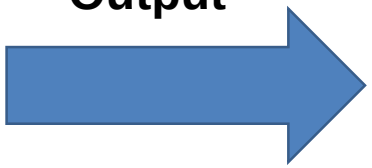
Operator	Description	Usage
:	Creates series of numbers from left operand to right operand	a:b
%in%	Identifies if an element(a) belongs to a vector(b)	a %in% b
%*%	Performs multiplication of a vector with its transpose	A %*% t(A)

R Programming; Misc. operators

```
# R Operators - R Misc Operators

a = 23:31
print ( a )

a = c(25, 27, 76)
b = 27
print ( b %in% a )
```

```
M = matrix(c(1,2,3,4), 2, 2, TRUE)
print ( M %*% t(M) )
```

Output



```
[1] 23 24 25 26 27 28 29 30 31
[1] TRUE
     [,1] [,2]
[1,]    5   11
[2,]   11   25
```

R Programming; R-Objects

- The frequently used **R-Objects**:
  - Vectors
  - Lists
  - Matrices
  - Arrays
  - Factors
  - Dataframes

R Programming; Matrix

- The frequently used **R-Objects**:
  - Vectors
  - Lists
  - Matrices
  - Arrays
  - Factors
  - Dataframes

R Programming; Matrix

What is a matrix?

Matrices or more generally arrays are multi-dimensional generalizations of vectors. In fact, they are vectors that can be indexed by two or more indices and will be printed in special ways.

A rudimentary knowledge of linear algebra is essential for using R and its powerful matrix operations.

How to create blank matrix, initialise and access to the elements of a matrix?

R Programming; Matrix

How to create blank matrix, initialise and access to the elements of a matrix?

```
Empty_matrix <- matrix(nrow = 3, ncol = 5) # logical matrix
Empty_matrix <- matrix(data = NA , nrow = 3, ncol = 5)

print("Empty matrix of 3 rows and 5 columns:") # with NA
print(Empty_matrix)
```

	V1	V2	V3	V4	V5
1	*NA*	*NA*	*NA*	*NA*	*NA*
2	*NA*	*NA*	*NA*	*NA*	*NA*
3	*NA*	*NA*	*NA*	*NA*	*NA*

How to view this matrix in RStudio without using print()?

R Programming; Matrix

How to create blank matrix, initialise and access to the elements of a matrix?

How to view this matrix in RStudio without using print()?

```
View(Empty_matrix)
```

	[, 1]	[, 2]	[, 3]	[, 4]	[, 5]
[1,]	NA	NA	NA	NA	NA
[2,]	NA	NA	NA	NA	NA
[3,]	NA	NA	NA	NA	NA

```
class(Empty_matrix)
```
→ "matrix"

```
attributes(Empty_matrix)
```
→
```
$dim
[1] 3 5
```

R Programming; Matrix

How to create blank matrix, initialise and access to the elements of a matrix?

Create a matrix taking a given vector of numbers:

```
int_Matrix <- matrix(c(1:16), nrow = 4, byrow = TRUE) # int matrix
print(int_Matrix)
```

	[, 1]	[, 2]	[, 3]	[, 4]
[1,]	1	2	3	4
[2,]	5	6	7	8
[3,]	9	10	11	12
[4,]	13	14	15	16

R Programming; Matrix

How to create blank matrix, initialise and access to the elements of a matrix?

Another example:

```
# double matrix
double_Matrix <- matrix(data = 0, nrow = 4, ncol = 4)
print(double_Matrix)
```

	[, 1]	[, 2]	[, 3]	[, 4]
[1,]	0	0	0	0
[2,]	0	0	0	0
[3,]	0	0	0	0
[4,]	0	0	0	0

```
typeof(double_Matrix)
```

“double”

```
dim(double_Matrix)
```

[1] 4 4

R Programming; Matrix

How to create blank matrix, initialise and access to the elements of a matrix?

	[, 1]	[, 2]	[, 3]	[, 4]
[1,]	0	0	0	0
[2,]	0	0	0	0
[3,]	0	0	0	0
[4,]	0	0	0	0

```
# Access to an element of this matrix located
# in row 2 and column 4

row2_col4_value <- double_Matrix[2,4]   # row2_col4_value = 0
double_Matrix[2,4] <- 67.89    # modify a single element of the matrix
```

R Programming; Matrix

How to create blank matrix, initialise and access to the elements of a matrix?

	[, 1]	[, 2]	[, 3]	[, 4]
[1,]	0	0	0	0
[2,]	0	0	0	67.89
[3,]	0	0	0	0
[4,]	0	0	0	0

```
# Access to an element of this matrix located
# in row 2 and column 4

row2_col4_value <- double_Matrix[2,4]
double_Matrix[2,4] <- 67.89
```

R Programming; Matrix

How to create blank matrix, initialise and access to the elements of a matrix?

```
# Access to different rows or columns
print(int_Matrix[2,]) # print all the elements of the second row
print(int_Matrix[,4]) # print all the elements of the fourth column
print(int_Matrix[c(1,2),c(3,4)])    #
print(int_Matrix[-c(1,2),])    #
```

	[, 1]	[, 2]	[, 3]	[, 4]
[1,]	1	2	3	4
[2,]	5	6	7	8
[3,]	9	10	11	12
[4,]	13	14	15	16

```
dim(double_Matrix)
```

```
[1] 4 4
```

```
(dim(double_Matrix))[1]

(dim(double_Matrix))[2]
```

R Programming; Matrix

How to create blank matrix, initialise and access to the elements of a matrix?

```
# Access to different rows or columns
print(int_Matrix[2,]) # print all the elements of the second row
print(int_Matrix[,4]) # print all the elements of the fourth column
print(int_Matrix[c(1,2),c(3,4)])    #
print(int_Matrix[-c(1,2),])   #
```

	[, 1]	[, 2]	[, 3]	[, 4]
[1,]	1	2	3	4
[2,]	5	6	7	8
[3,]	9	10	11	12
[4,]	13	14	15	16

R Programming; Matrix

How to create blank matrix, initialise and access to the elements of a matrix?

```
# Access to different rows or columns
print(int_Matrix[2,]) # print all the elements of the second row
print(int_Matrix[,4]) # print all the elements of the fourth column
print(int_Matrix[c(1,2),c(3,4)])    #
print(int_Matrix[-c(1,2),])   #
```

	[, 1]	[, 2]	[, 3]	[, 4]
[1,]	1	2	3	4
[2,]	5	6	7	8
[3,]	9	10	11	12
[4,]	13	14	15	16

R Programming; Matrix

How to create blank matrix, initialise and access to the elements of a matrix?

```
# Access to different rows or columns
print(int_Matrix[2,]) # print all the elements of the second row
print(int_Matrix[,4]) # print all the elements of the fourth column
print(int_Matrix[c(1,2),c(3,4)])    #
print(int_Matrix[-c(1,2),])    #
```

	[, 1]	[, 2]	[, 3]	[, 4]
[1,]	1	2	3	4
[2,]	5	6	7	8
[3,]	9	10	11	12
[4,]	13	14	15	16

R Programming; Matrix

How to create blank matrix, initialise and access to the elements of a matrix?

```
# Access to different rows or columns
print(int_Matrix[2,]) # print all the elements of the second row
print(int_Matrix[,4]) # print all the elements of the fourth column
print(int_Matrix[c(1,2),c(3,4)])    #
print(int_Matrix[-c(1,2),])    #
```

	[, 1]	[, 2]	[, 3]	[, 4]
[1,]	1	2	3	4
[2,]	5	6	7	8
[3,]	9	10	11	12
[4,]	13	14	15	16

R Programming; Matrix

How to create blank matrix, initialise and access to the elements of a matrix?

```
# Access to different rows or columns
print(int_Matrix[2,]) # print all the elements of the second row
print(int_Matrix[,4]) # print all the elements of the fourth column
print(int_Matrix[c(1,2),c(3,4)])    #
print(int_Matrix[-c(1,2),])    #
```

	[, 1]	[, 2]	[, 3]	[, 4]
[1,]	1	2	3	4
[2,]	5	6	7	8
[3,]	9	10	11	12
[4,]	13	14	15	16

```
dim(double_Matrix)
```

```
[1] 4 4
```

```
(dim(double_Matrix))[1]

(dim(double_Matrix))[2]
```

R Programming; Matrix

How to create blank matrix, initialise and access to the elements of a matrix?

```
# Access based on a condition
print(int_Matrix[int_Matrix > 10]) #
```

	[, 1]	[, 2]	[, 3]	[, 4]
[1,]	1	2	3	4
[2,]	5	6	7	8
[3,]	9	10	11	12
[4,]	13	14	15	16

Which one of the following is the correct answer?

```
[1] 11  12  13  14  15  16
[1] 13  14  11  15  12  16
```

?

R Programming; Matrix

How to create blank matrix, initialise and access to the elements of a matrix?

```
# Access based on a condition
print(int_Matrix[int_Matrix > 10]) #
```

	[, 1]	[, 2]	[, 3]	[, 4]
[1,]	1	2	3	4
[2,]	5	6	7	8
[3,]	9	10	11	12
[4,]	13	14	15	16

Which one of the following is the correct answer?

```
[1] 11  12  13  14  15  16
[1] 13  14  11  15  12  16
```

Column based

R Programming; Matrix

How to create blank matrix, initialise and access to the elements of a matrix?

```
# Access/modify based on a condition
int_Matrix[int_Matrix > 10] <- -1
```

	[, 1]	[, 2]	[, 3]	[, 4]
[1,]	1	2	3	4
[2,]	5	6	7	8
[3,]	9	10	11	12
[4,]	13	14	15	16

R Programming; Matrix

How to create blank matrix, initialise and access to the elements of a matrix?

```
# Access/modify based on a condition
int_Matrix[int_Matrix > 10] <- -1
```

	[, 1]	[, 2]	[, 3]	[, 4]
[1,]	1	2	3	4
[2,]	5	6	7	8
[3,]	9	10	-1	-1
[4,]	-1	-1	-1	-1

R Programming; Matrix

Transpose of a matrix

```
# Transpose a matrix
int_Matrix <- t(int_Matrix)
```

	[, 1]	[, 2]	[, 3]	[, 4]
[1,]	1	2	3	4
[2,]	5	6	7	8
[3,]	9	10	-1	-1
[4,]	-1	-1	-1	-1

R Programming; Matrix

Transpose of a matrix

```
# Transpose a matrix
int_Matrix <- t(int_Matrix)  # switches the row and column indices
```

	[, 1]	[, 2]	[, 3]	[, 4]
[1,]	1	2	3	4
[2,]	5	6	7	8
[3,]	9	10	-1	-1
[4,]	-1	-1	-1	-1

	[, 1]	[, 2]	[, 3]	[, 4]
[1,]	**1**	**5**	**9**	**-1**
[2,]	**2**	**6**	**10**	**-1**
[3,]	**3**	**7**	**-1**	**-1**
[4,]	**4**	**8**	**-1**	**-1**

Transposed matrix

R Programming; Matrix

How to create a matrix using cbind() or rbind()?

```
# Create a matrix using cbind()
int_Matrix <- cbind(c(1:3),c(7:9),c(4,6,10)) # it binds three columns
```

	[, 1]	[, 2]	[, 3]
[1,]	1	7	4
[2,]	2	8	6
[3,]	3	9	10

```
# Create a matrix using rbind()
int_Matrix <- rbind(c(1,7,4),c(2,8,6),c(3,9,10)) # it binds three rows
```

	[, 1]	[, 2]	[, 3]
[1,]	1	7	4
[2,]	2	8	6
[3,]	3	9	10

R Programming; Matrix

How to create blank matrix, initialise and access to the elements of a matrix?

```
Empty_matrix <- matrix(nrow = 3, ncol = 5) # logical matrix
Empty_matrix <- matrix(data = NA , nrow = 3, ncol = 5)
```

	V1	V2	V3	V4	V5
1	*NA*	*NA*	*NA*	*NA*	*NA*
2	*NA*	*NA*	*NA*	*NA*	*NA*
3	*NA*	*NA*	*NA*	*NA*	*NA*

Default names

How to change the names of columns rows?

	Sample_1	Sample_2	Sample_3	Sample_4	Sample_5
Feature_1	*NA*	*NA*	*NA*	*NA*	*NA*
Feature_2	*NA*	*NA*	*NA*	*NA*	*NA*
Feature_3	*NA*	*NA*	*NA*	*NA*	*NA*

R Programming; Matrix

How to create blank matrix, initialise and access to the elements of a matrix?

```
Empty_matrix <- matrix(nrow = 3, ncol = 5) # logical matrix
Empty_matrix <- matrix(data = NA , nrow = 3, ncol = 5)
```

	V1	V2	V3	V4	V5
1	*NA*	*NA*	*NA*	*NA*	*NA*
2	*NA*	*NA*	*NA*	*NA*	*NA*
3	*NA*	*NA*	*NA*	*NA*	*NA*

Default names

How to change the column (row) names?

	Sample_1	Sample_2	Sample_3	Sample_4	Sample_5
Feature_1	*NA*	*NA*	*NA*	*NA*	*NA*
Feature_2	*NA*	*NA*	*NA*	*NA*	*NA*
Feature_3	*NA*	*NA*	*NA*	*NA*	*NA*

```
colnames(Empty_matrix) <- c("Sample_1", "Sample_2", "Sample_3", "Sample_4", "Sample_5")
```

R Programming; Matrix

How to create blank matrix, initialise and access to the elements of a matrix?

```
Empty_matrix <- matrix(nrow = 3, ncol = 5) # logical matrix
Empty_matrix <- matrix(data = NA , nrow = 3, ncol = 5)
```

	V1	V2	V3	V4	V5
1	*NA*	*NA*	*NA*	*NA*	*NA*
2	*NA*	*NA*	*NA*	*NA*	*NA*
3	*NA*	*NA*	*NA*	*NA*	*NA*

Default names

How to change the column (row) names?

	Sample_1	Sample_2	Sample_3	Sample_4	Sample_5
Feature_1	*NA*	*NA*	*NA*	*NA*	*NA*
Feature_2	*NA*	*NA*	*NA*	*NA*	*NA*
Feature_3	*NA*	*NA*	*NA*	*NA*	*NA*

```
rownames(Empty_matrix) <- c("Feature_1", "Feature_2", "Feature_3")
```

R Programming; Misc. operators

```
# R Operators - R Misc Operators

a = 23:31
print ( a )

a = c(25, 27, 76)
b = 27
print ( b %in% a )

M = matrix(c(1,2,3,4), 2, 2, TRUE)
print ( M %*% t(M) )
```

Output

```
[1] 23 24 25 26 27 28 29 30 31
[1] TRUE
     [,1] [,2]
[1,]    5   11
[2,]   11   25
```

M

	V1	V2
1	1	2
2	3	4

```
matrix(data=c(1,2,3,4), nrow=2, ncol=2, byrow= TRUE)
```

R Programming; R-Objects

- The frequently used **R-Objects**:
  - Vectors
  - Lists
  - Matrices
  - Arrays
  - Factors
  - Dataframes