R Syntax 1: Data Types and Vector

Pilsung Kang
School of Industrial Management Engineering
Korea University

Basic Instructions

- Getting Help, Using Packages, and Working Directory

Getting Help

Accessing the help files

`?mean`
Get help of a particular function.

`help.search('weighted mean')`
Search the help files for a word or phrase.

`help(package = 'dplyr')`
Find help for a package.

More about an object

`str(iris)`
Get a summary of an object's structure.

`class(iris)`
Find the class an object belongs to.

Using Packages

`install.packages('dplyr')`
Download and install a package from CRAN.

`library(dplyr)`
Load the package into the session, making all its functions available to use.

`dplyr::select`
Use a particular function from a package.

`data(iris)`
Load a built-in dataset into the environment.

Working Directory

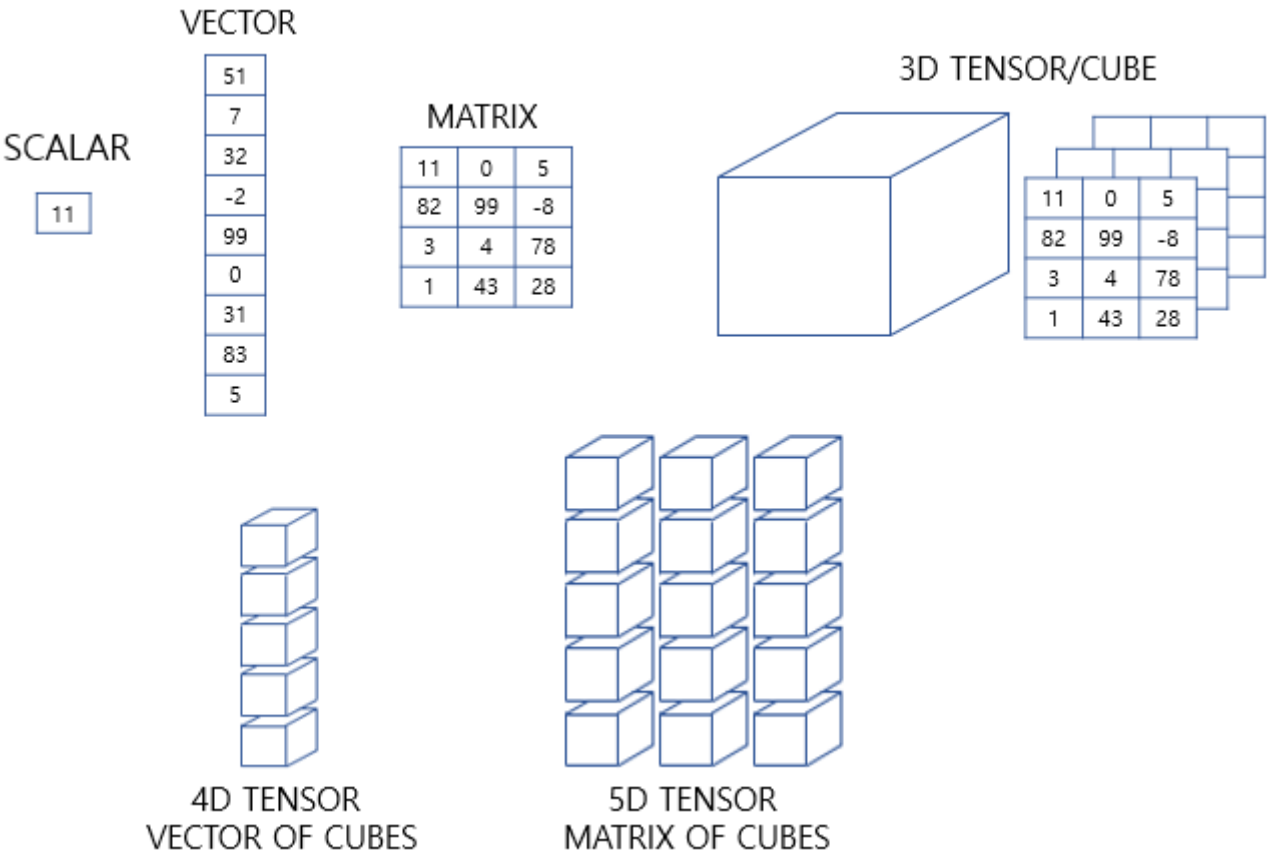
`getwd()`
Find the current working directory (where inputs are found and outputs are sent).

`setwd('C://file/path')`
Change the current working directory.

Use projects in RStudio to set the working directory to the folder you are working in.

Data Types in R

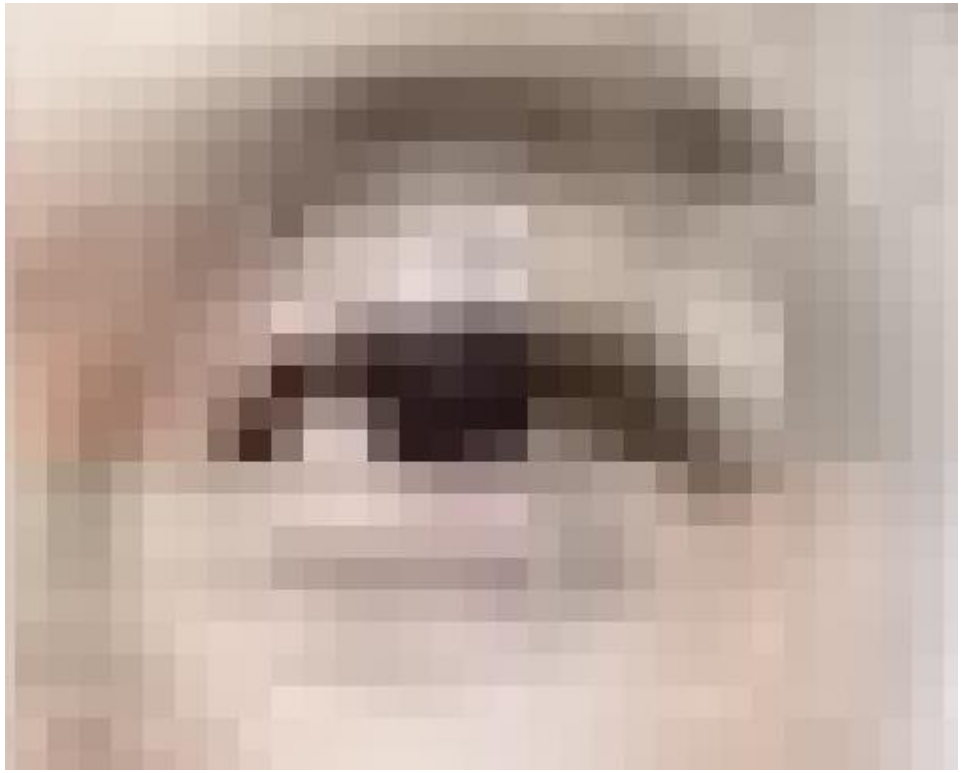
- Data Types w.r.t. dimensions



https://wikidocs.net/37001

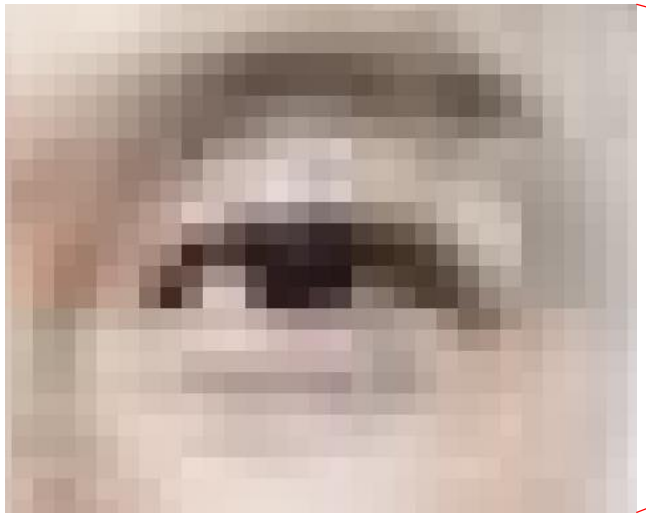
Data Types in R

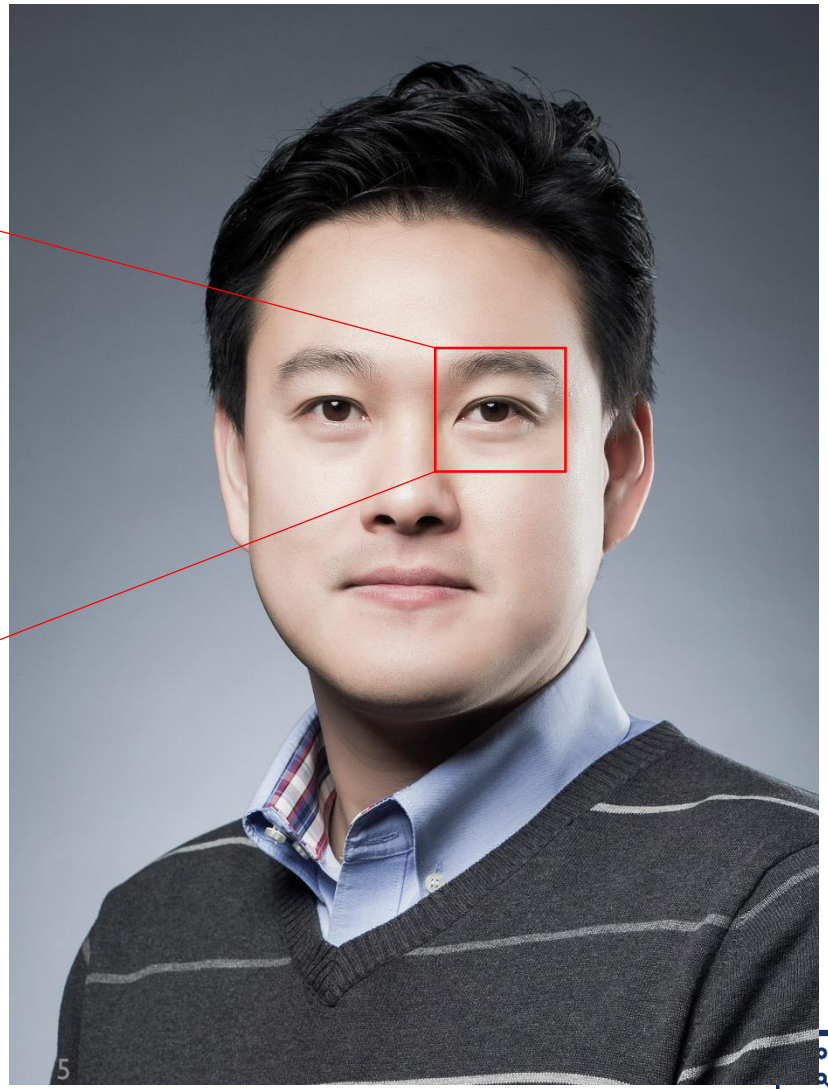
- Tensor in Data Analytics
  - ✓ Whose eye is it?



Data Types in R

- Tensor in Data Analytics
  - ✓ Whose eye is it?
    - It's my eye





고려대학교 KOREA UNIVERSITY
DSBA Data Science & Business Analytics

Data Types in R

- Computers recognize an image as a 3-D Tensor: Width X Height X 3 (RGB)

1,685

2,247

```
> pskang[1:10, 1:10, 1]
           [,1]      [,2]      [,3]      [,4]      [,5]      [,6]      [,7]      [,8]      [,9]     [,10]
 [1,] 0.2784314 0.2784314 0.2745098 0.2745098 0.2745098 0.2745098 0.2705882 0.2705882 0.2862745 0.2901961
 [2,] 0.2745098 0.2745098 0.2745098 0.2745098 0.2745098 0.2745098 0.2745098 0.2745098 0.2666667 0.2705882
 [3,] 0.2745098 0.2745098 0.2745098 0.2745098 0.2745098 0.2745098 0.2745098 0.2745098 0.2705882 0.2745098
 [4,] 0.2705882 0.2705882 0.2745098 0.2745098 0.2745098 0.2745098 0.2784314 0.2784314 0.2823529 0.2823529
 [5,] 0.2705882 0.2705882 0.2705882 0.2745098 0.2745098 0.2784314 0.2784314 0.2784314 0.2784314 0.2784314
 [6,] 0.2705882 0.2705882 0.2745098 0.2745098 0.2745098 0.2745098 0.2784314 0.2784314 0.2823529 0.2784314
 [7,] 0.2705882 0.2745098 0.2745098 0.2745098 0.2745098 0.2745098 0.2745098 0.2784314 0.2941176 0.2862745
 [8,] 0.2745098 0.2745098 0.2745098 0.2745098 0.2745098 0.2745098 0.2745098 0.2745098 0.2901961 0.2823529
 [9,] 0.2745098 0.2705882 0.2745098 0.2784314 0.2823529 0.2823529 0.2745098 0.2666667 0.2784314 0.2784314
[10,] 0.2784314 0.2784314 0.2784314 0.2823529 0.2862745 0.2862745 0.2784314 0.2745098 0.2745098 0.2745098
```

```
> pskang[1:10, 1:10, 2]
           [,1]      [,2]      [,3]      [,4]      [,5]      [,6]      [,7]      [,8]      [,9]     [,10]
 [1,] 0.2980392 0.2980392 0.2941176 0.2941176 0.2941176 0.2941176 0.2901961 0.2901961 0.2980392 0.3019608
 [2,] 0.2941176 0.2941176 0.2941176 0.2941176 0.2941176 0.2941176 0.2941176 0.2941176 0.2784314 0.2823529
 [3,] 0.2941176 0.2941176 0.2941176 0.2941176 0.2941176 0.2941176 0.2941176 0.2941176 0.2823529 0.2862745
 [4,] 0.2901961 0.2901961 0.2941176 0.2941176 0.2941176 0.2941176 0.2980392 0.2980392 0.2941176 0.2941176
 [5,] 0.2901961 0.2901961 0.2901961 0.2941176 0.2941176 0.2980392 0.2980392 0.2980392 0.2901961 0.2901961
 [6,] 0.2901961 0.2901961 0.2941176 0.2941176 0.2941176 0.2941176 0.2980392 0.2980392 0.2941176 0.2901961
 [7,] 0.2901961 0.2941176 0.2941176 0.2941176 0.2941176 0.2941176 0.2941176 0.2980392 0.3058824 0.2980392
 [8,] 0.2941176 0.2941176 0.2941176 0.2941176 0.2941176 0.2941176 0.2941176 0.2941176 0.3019608 0.2941176
 [9,] 0.2862745 0.2823529 0.2862745 0.2901961 0.2941176 0.2941176 0.2862745 0.2784314 0.2901961 0.2901961
[10,] 0.2901961 0.2901961 0.2901961 0.2941176 0.2980392 0.2980392 0.2901961 0.2862745 0.2862745 0.2862745
```

```
> pskang[1:10, 1:10, 3]
           [,1]      [,2]      [,3]      [,4]      [,5]      [,6]      [,7]      [,8]      [,9]     [,10]
 [1,] 0.3215686 0.3215686 0.3176471 0.3176471 0.3176471 0.3176471 0.3137255 0.3137255 0.3254902 0.3294118
 [2,] 0.3176471 0.3176471 0.3176471 0.3176471 0.3176471 0.3176471 0.3176471 0.3176471 0.3058824 0.3098039
 [3,] 0.3176471 0.3176471 0.3176471 0.3176471 0.3176471 0.3176471 0.3176471 0.3176471 0.3098039 0.3137255
 [4,] 0.3137255 0.3137255 0.3176471 0.3176471 0.3176471 0.3176471 0.3215686 0.3215686 0.3215686 0.3215686
 [5,] 0.3137255 0.3137255 0.3137255 0.3176471 0.3176471 0.3215686 0.3215686 0.3215686 0.3176471 0.3176471
 [6,] 0.3137255 0.3137255 0.3176471 0.3176471 0.3176471 0.3176471 0.3215686 0.3215686 0.3215686 0.3176471
 [7,] 0.3137255 0.3176471 0.3176471 0.3176471 0.3176471 0.3176471 0.3176471 0.3215686 0.3333333 0.3254902
 [8,] 0.3176471 0.3176471 0.3176471 0.3176471 0.3176471 0.3176471 0.3176471 0.3176471 0.3294118 0.3215686
 [9,] 0.3058824 0.3019608 0.3058824 0.3098039 0.3137255 0.3137255 0.3058824 0.2980392 0.3098039 0.3098039
[10,] 0.3098039 0.3098039 0.3098039 0.3137255 0.3176471 0.3176471 0.3098039 0.3058824 0.3058824 0.3058824
```

Data Types in R

- Variable types
  - ✓ Homogeneous variables
    - All elements are the same type: numeric values in this example

	Year	January	February	March	April	May	June	July	August	September	October	November	December
1	1998	0	0	2	21	47	272	391	262	251	178	47	8
2	1999	0	4	1	24	145	230	448	195	117	248	17	2
3	2000	3	9	0	28	74	281	309	341	190	169	10	8
4	2001	1	1	1	64	42	245	271	233	177	127	30	2
5	2002	2	12	2	24	87	179	107	173	80	178	18	1
6	2003	0	2	18	37	7	182	205	172	72	166	9	1
7	2004	2	1	6	54	178	202	201	193	140	97	22	0
8	2005	5	2	3	67	57	239	472	295	210	196	35	3
9	2006	0	0	26	17	152	270	356	273	168	74	67	0
10	2007	0	0	1	36	62	344	371	350	270	118	19	8
11	2008	0	15	129	33	61	172	189	262	161	106	37	2

Data Types in R

- Variable types
  - ✓ Homogeneous variables
    - Variables (Columns) are different types

	Name	Number	Position	Age	Height	Weight	College	Salary
0	Avery Bradley	0.0	PG	25.0	6-2	180.0	Texas	7730337.0
1	Jae Crowder	99.0	SF	25.0	6-6	235.0	Marquette	6796117.0
2	John Holland	30.0	SG	27.0	6-5	205.0	Boston University	NaN
3	R.J. Hunter	28.0	SG	22.0	6-5	185.0	Georgia State	1148640.0
4	Jonas Jerebko	8.0	PF	29.0	6-10	231.0	NaN	5000000.0
5	Amir Johnson	90.0	PF	29.0	6-9	240.0	NaN	12000000.0
6	Jordan Mickey	55.0	PF	21.0	6-8	235.0	LSU	1170960.0
7	Kelly Olynyk	41.0	C	25.0	7-0	238.0	Gonzaga	2165160.0
8	Terry Rozier	12.0	PG	22.0	6-2	190.0	Louisville	1824360.0
9	Marcus Smart	36.0	PG	22.0	6-4	220.0	Oklahoma State	3431040.0

Data Types in R

- Questions
  - ✓ Q1: Are all variables homogeneous?
  - ✓ Q2: Are there more than one record?

Attribute\No. Records	1	>= 2
Homogeneous	Vector	Matrix or Array
Heterogeneous	List	Dataframe

- Dataframe makes R powerful to analyze heterogeneous multivariate data

Data Types in R

Scalar	Vector	List	Matrix	Array	Factor	Data.frame

- Vector
  - ✓ Vectors are homogeneous
    - All elements in a vector should be the same mode
  - ✓ Vector has an index for each element
    - A set of indices returns the corresponding sub-vector
    - Index starts from 1 (python: 0)
  - ✓ The elements of a vector can have its own name
  - ✓ Vectors in R is a column-wise vectors

```r
# Part 1-1: Data Handling (Vector) ----------

# Assign values to the vector A & B
A <- c(1,2,3)
B <- c(1, "A", 0.5)

# Check the mode
mode(A)
mode(B)

# Select a subset of vector
A[1]
A[2:3]
A[c(2,3)]

# Assign names
names(A)
names(A) <- c("First", "Second", "Third")

# call by index or name
A[1]
A["First"]
```

Handling Vectors

- Vector initiation
  - ✓ Do not have to initiate → creation and value assignment are done at the same time
    - a <- 3: create a vector named 'a' and assign the value 3 to it
- Add elements to an existing vector
  - ✓ The size of a vector is fixed when it is created
  - ✓ We have to recreate the vector if we want to add or remove some elements

```
24  # Data Handling: Vector
25  x <- c(1,2,3,4)
26  x
27  x <- c(x[1:3], 10, x[4])
28  x
29  length(x)
```

Handling Vectors

- Vector reuse
  - ✓ When R conduct an operation with two vectors, the shorter vector is reused to avoid an error

```
> c(1,2,4) + c(10,11,12,13,14)
[1] 11 13 16 14 16
Warning message:
In c(1, 2, 4) + c(10, 11, 12, 13, 14) :
  longer object length is not a multiple of shorter object length
```

  - ✓ Column-first

```
> x <- matrix(1:6, nrow=3, ncol=2)
> x
     [,1] [,2]
[1,]    1    4
[2,]    2    5
[3,]    3    6
> x + c(1:2)
     [,1] [,2]
[1,]    2    6
[2,]    4    6
[3,]    4    8
```

Arithmetic Operators

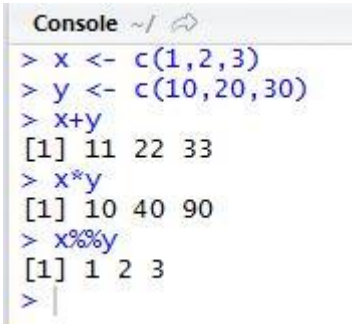
Operator	Description
+	addition
-	subtraction
*	multiplication
/	division
^ or **	exponentiation
x %% y	modulus (x mod y) 5%%2 is 1
x %/% y	integer division 5%/%2 is 2

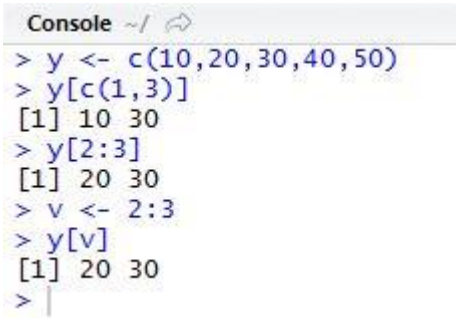
Logical Operators

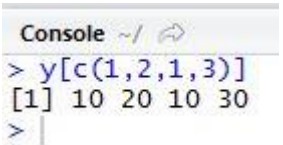
Operator	Description
<	less than
<=	less than or equal to
>	greater than
>=	greater than or equal to
==	exactly equal to
!=	not equal to
!x	Not x
x | y	x OR y
x & y	x AND y
isTRUE(x)	test if X is TRUE

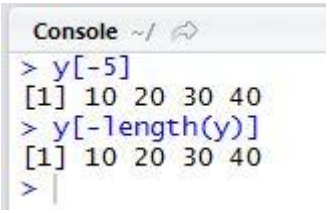
Handling Vectors

- Vector operations are element-wise
- Vector indexing
  - ✓ Extract a subset of vectors
  - ✓ Index can be used redundantly
  - ✓ A negative index is used to remove the corresponding element









Handling Vectors

- Creating vectors with operators
  - ✓ : operator: create vectors with certain range
  - ✓ seq: a generalized version of ":" operator
  - ✓ rep: repeat values

Operator Syntax and Precedence

Description

Outlines R syntax and gives the precedence of operators.

Details

The following unary and binary operators are defined. They are listed in precedence groups, from highest to lowest.

:: :::	access variables in a namespace
$ @	component / slot extraction
[[[	indexing
^	exponentiation (right to left)
- +	unary minus and plus
:	sequence operator
%any%	special operators (including %% and %/%)
* /	multiply, divide
+ -	(binary) add, subtract
< > <= >= == !=	ordering and comparison
!	negation
& &&	and
| ||	or
~	as in formulae
-> ->>	rightwards assignment
<- <<-	assignment (right to left)
=	assignment (right to left)
?	help (unary and binary)

Console ~/

```
> x <- 1:5
> y <- 5:1
> z <- 2
> 1:z-1
[1] 0 1
> 1:(z-1)
[1] 1
>
```

Console ~/

```
> seq(from=12,to=30,by=3)
[1] 12 15 18 21 24 27 30
> seq(from=12,to=30,by=4)
[1] 12 16 20 24 28
> seq(from=1.1,to=2,length=10)
 [1] 1.1 1.2 1.3 1.4 1.5 1.6 1.7 1.8 1.9 2.0
>
```

Console ~/

```
> rep(10,5)
[1] 10 10 10 10 10
> rep(c(10,20,30),3)
[1] 10 20 30 10 20 30 10 20 30
> rep(1:3,3)
[1] 1 2 3 1 2 3 1 2 3
> rep(c(10,20,30),each=3)
[1] 10 10 10 20 20 20 30 30 30
>
```

Handling Vectors

- Apply conditions for each element in a vector
  - ✓ any() function: return TRUE if at least one of the elements satisfies the condidition
  - ✓ all() function: return TRUE only when all elements satisfy the condition
- NA vs NULL
  - ✓ NA (Not Available): Some value exists but we cannot exactly know the value
  - ✓ NULL: Physically not exist

```
Console ~/
> x <- 1:10
> x > 8
 [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE  TRUE
> any(x > 8)
[1] TRUE
> any(x > 20)
[1] FALSE
> all(x > 8)
[1] FALSE
> all(x > 0)
[1] TRUE
>
```

```
Console ~/
> x <- c(1,2,NA,4,5)
> y <- c(1,2,NULL,4,5)
> mean(x)
[1] NA
> mean(x, na.rm = TRUE)
[1] 3
> mean(y)
[1] 3
>
```

Handling Vectors

- Filtering: Extract the element that satisfy a given condition
  - ✓ Directly extract from index
  - ✓ subset(): return the values that satisfy the condition
  - ✓ which(): return the indices that satisfy the condition

```
> x <- c(10,20,NA,40,50)
> x[x>20]
[1] NA 40 50
> subset(x, x>20)
[1] 40 50
> which(x>20)
[1] 4 5
```


ANY
questions?