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; Lists

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

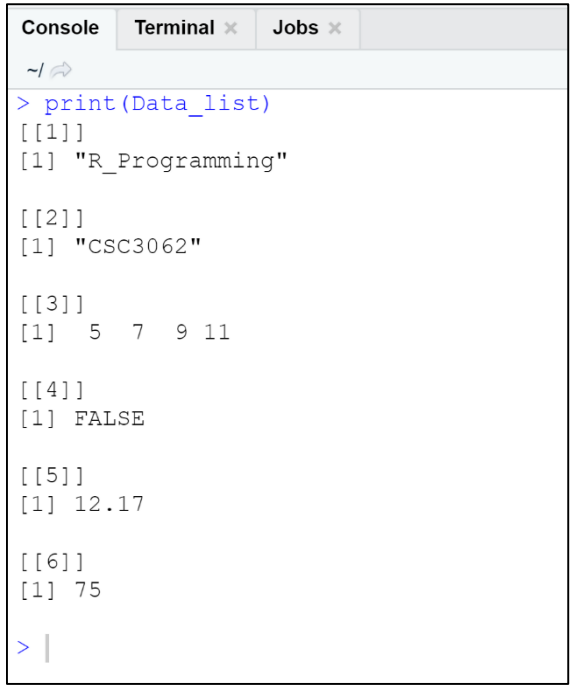
R Programming; Lists

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

R Programming; Lists

Lists are a bit like vectors, except that each entry can be any other R object, even another list.

```
Data_list <- list("R_Programming", "CSC3062", c(3, 5, 8, 13), FALSE, 12.17, 75)
print(Data_list)
```



Data_list[[1]] ← Access to the first element

Data_list[[2]]

Data_list[[3]]

Data_list[[4]]

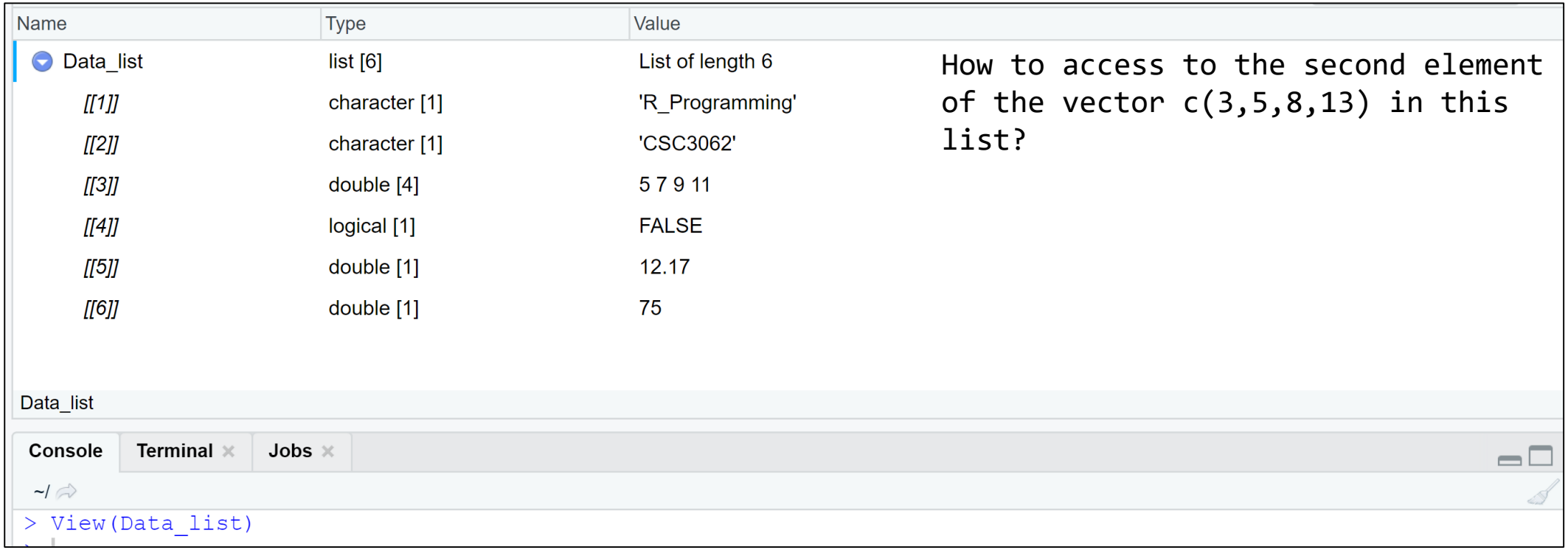
Data_list[[5]]

Data_list[[6]] ← Access to the sixth element

R Programming; Lists

Lists are a bit like vectors, except that each entry can be any other R object, even another list.

```
Data_list <- list("R_Programming", "CSC3062", c(3, 5, 8, 13), FALSE, 12.17, 75)
View(Data_list)
```

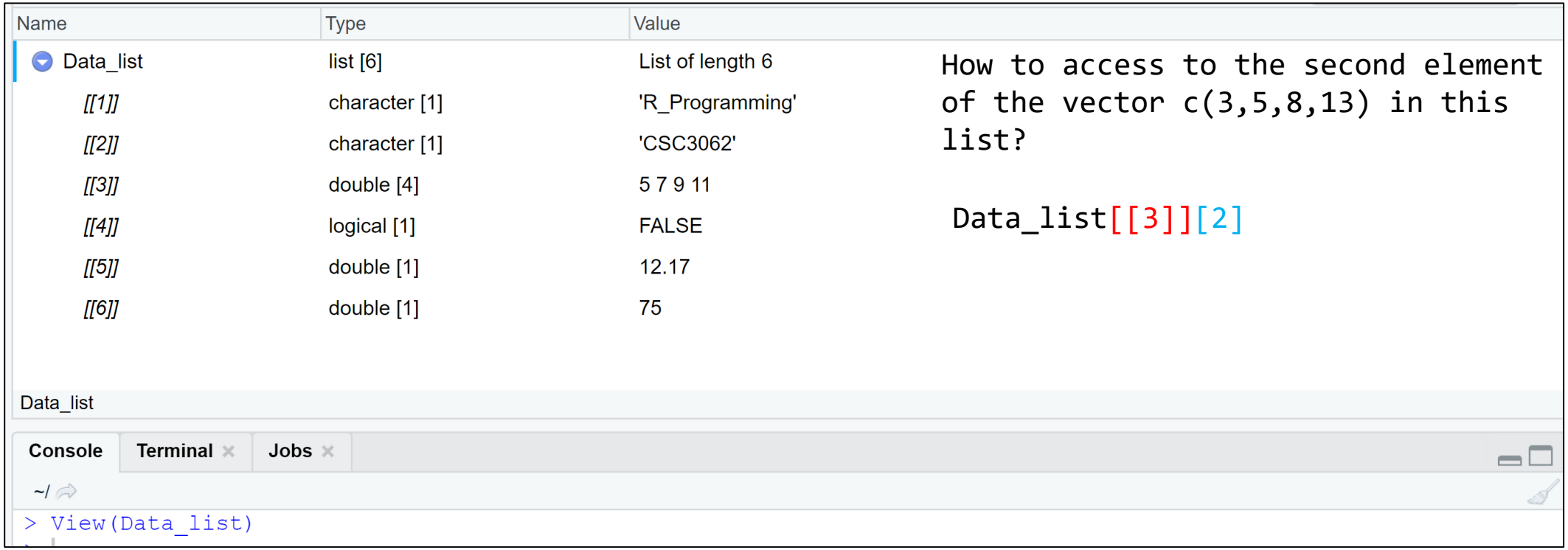


How to access to the second element of the vector c(3,5,8,13) in this list?

R Programming; Lists

Lists are a bit like vectors, except that each entry can be any other R object, even another list.

```
Data_list <- list("R_Programming", "CSC3062", c(3, 5, 8, 13), FALSE, 12.17, 75)
View(Data_list)
```



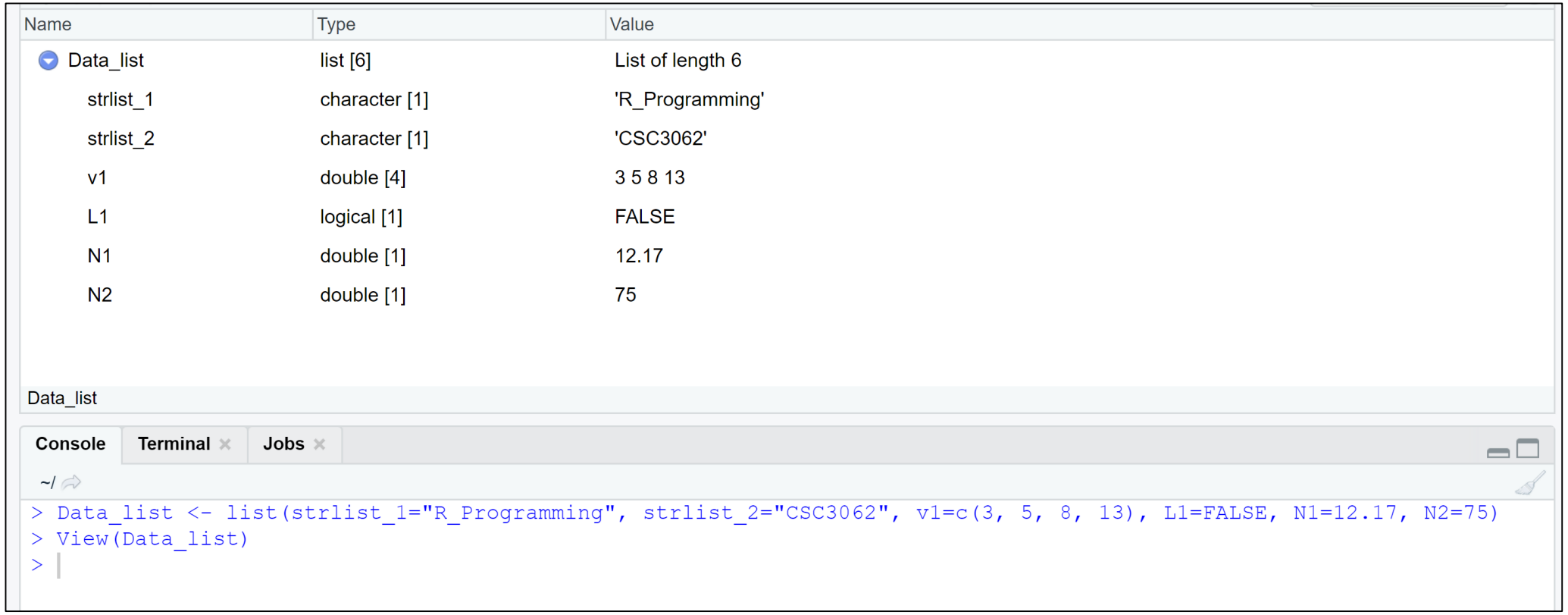
How to access to the second element of the vector c(3,5,8,13) in this list?

```
Data_list[[3]][2]
```

R Programming; Lists

Lists are a bit like vectors, except that each entry can be any other R object, even another list.

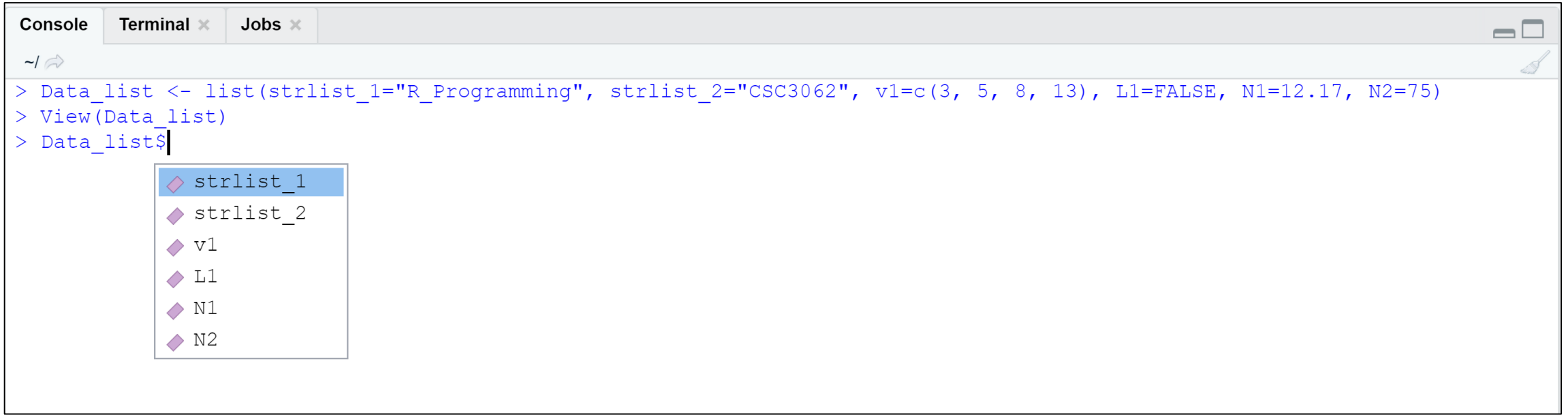
```
Data_list <- list(strlist_1="R_Programming", strlist_2="CSC3062", v1=c(3, 5, 8, 13), L1=FALSE, N1= 12.17, N2= 75)
```



R Programming; Lists

Lists are a bit like vectors, except that each entry can be any other R object, even another list.

```
Data_list <- list(strlist_1="R_Programming", strlist_2="CSC3062", v1=c(3, 5, 8, 13), L1=FALSE, N1= 12.17, N2= 75)
```



R Programming; Lists

Lists are a bit like vectors, except that each entry can be any other R object, even another list.

```
Data_list <- list(strlist_1="R_Programming", strlist_2="CSC3062", v1=c(3, 5, 8, 13), L1=FALSE, N1= 12.17, N2= 75)
```

Console | Terminal | Jobs

~/

```
> Data_list <- list(strlist_1="R_Programming", strlist_2="CSC3062", v1=c(3, 5, 8, 13), L1=FALSE, N1=12.17, N2=75)
> View(Data_list)
> Data_list$strlist_1
[1] "R_Programming"
> Data_list$strlist_2
[1] "CSC3062"
> Data_list$v1
[1]  3  5  8 13
> Data_list$v1[1:4]
[1]  3  5  8 13
> Data_list$L1
[1] FALSE
> Data_list$N1
[1] 12.17
> Data_list$N2
[1] 75
> names(Data_list)
[1] "strlist_1" "strlist_2" "v1" "L1" "N1" "N2"
```

R Programming; Lists

Lists are a bit like vectors, except that each entry can be any other R object, even another list.

```
n1 = list(1,2,3)
c1 = list("Red", "Green", "Black")
print("Original lists:")
print(n1)
print(c1)
print("Merge the said lists:")
merge_list = c(n1, c1)
print("New merged list:")
print(merge_list)
```

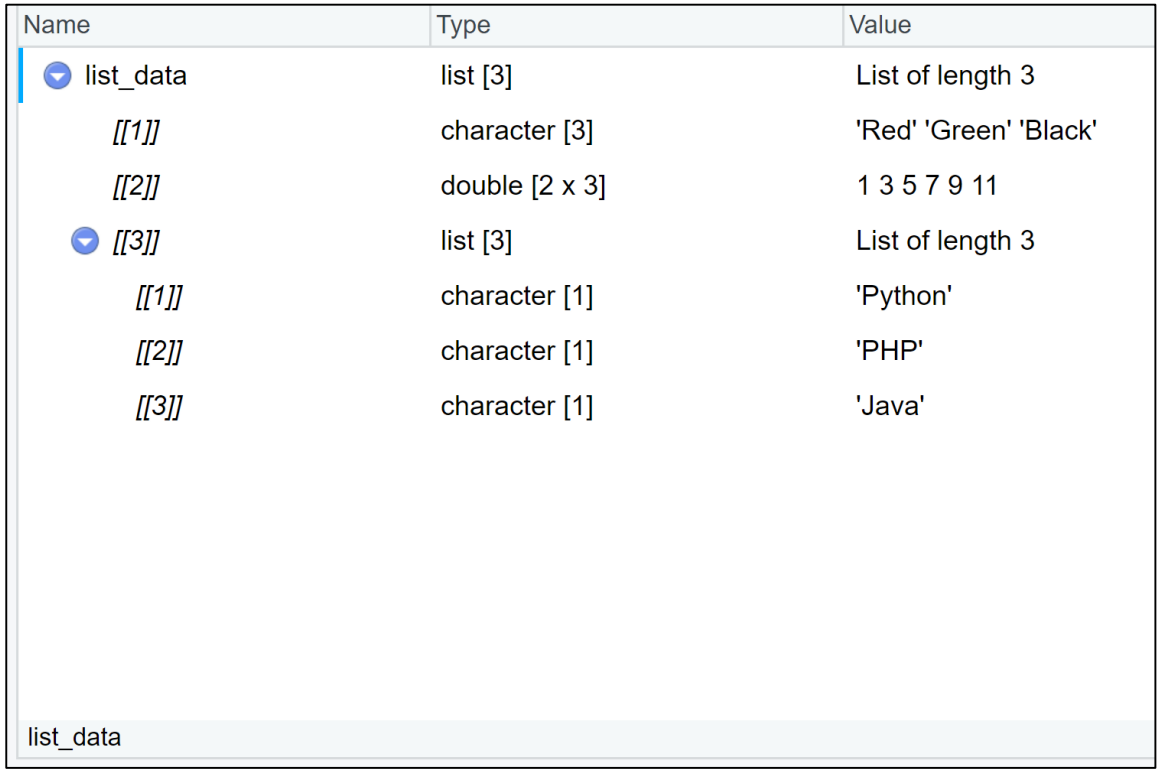
Name	Type	Value
merge_list	list [6]	List of length 6
[[1]]	double [1]	1
[[2]]	double [1]	2
[[3]]	double [1]	3
[[4]]	character [1]	'Red'
[[5]]	character [1]	'Green'
[[6]]	character [1]	'Black'

merge_list

R Programming; Lists

Lists are a bit like vectors, except that each entry can be any other R object, even another list.

```
list_data <- list(c("Red","Green","Black"), matrix(c(1,3,5,7,9,11), nrow = 2), list("Python", "PHP", "Java"))
print("List:")
print(list_data)
print("Remove the second element of the list:")
list_data[2] = NULL
print("New list:")
print(list_data)
```



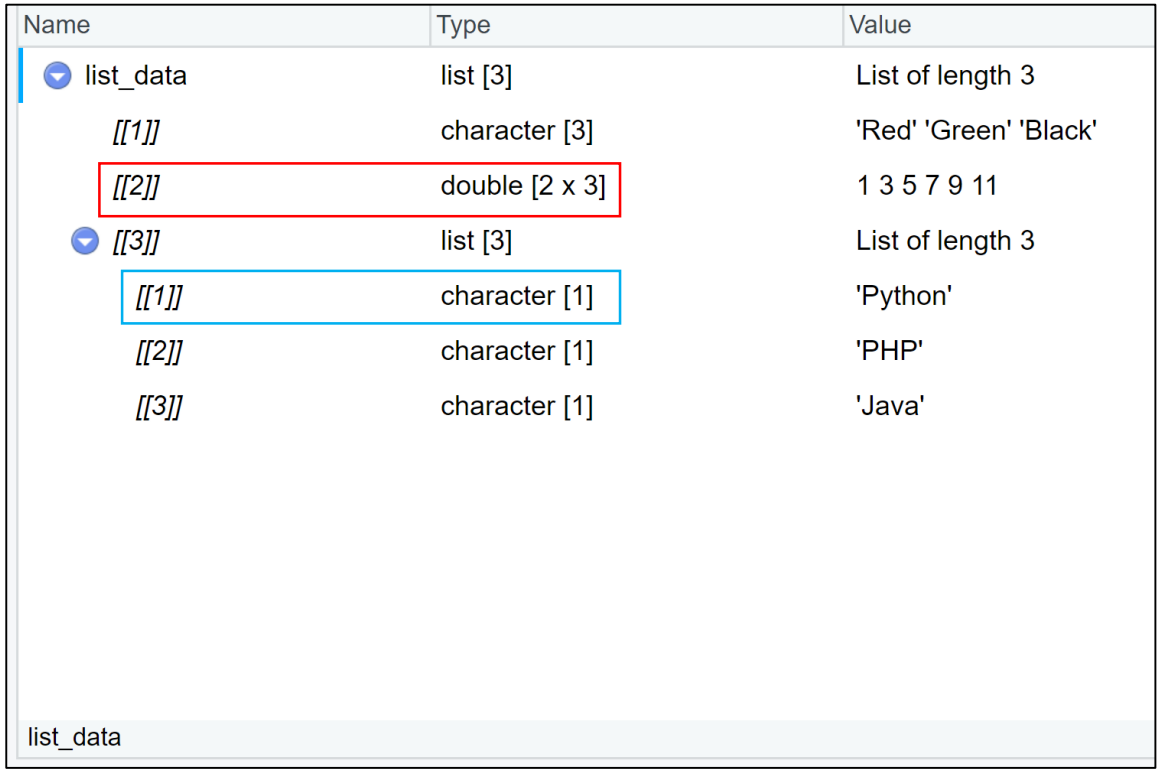

```
list_data[[2]][,]

list_data[[3]][[1]]
```

?

R Programming; Lists

Lists are a bit like vectors, except that each entry can be any other R object, even another list.

```
list_data <- list(c("Red","Green","Black"), matrix(c(1,3,5,7,9,11), nrow = 2), list("Python", "PHP", "Java"))
print("List:")
print(list_data)
print("Remove the second element of the list:")
list_data[2] = NULL
print("New list:")
print(list_data)
```




```
list_data[[2]][,]

list_data[[3]][[1]]
```

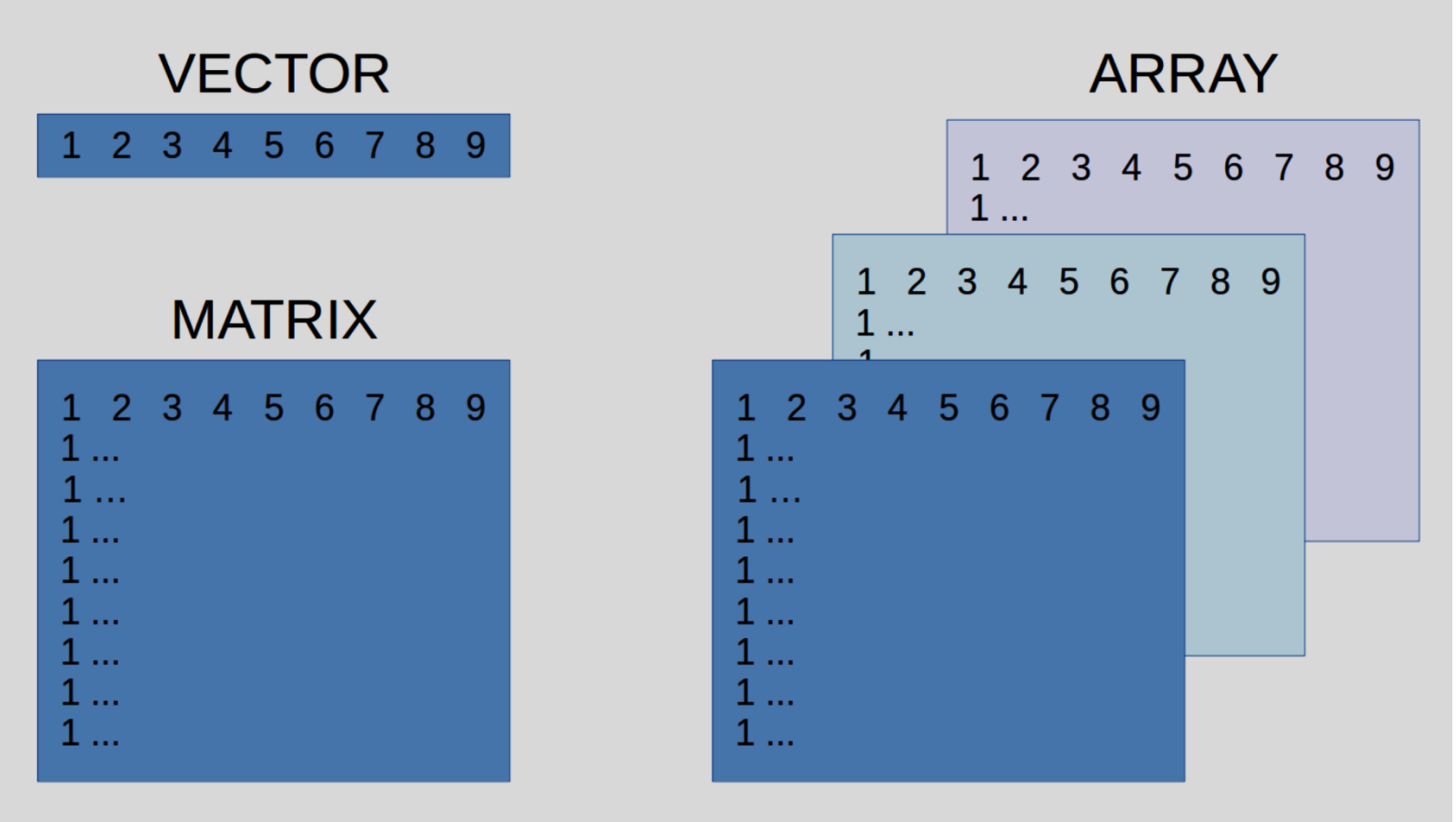
?

R Programming; Arrays

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

R Programming; Arrays

- An array is a **multi-dimensional collection of matrices**
- The generalization of matrices to higher dimensions is the array.
- Arrays are defined much like matrices, with a call to the **array()** command.



R Programming; Arrays

- An array is a **multi-dimensional collection of matrices**
- The generalization of matrices to higher dimensions is the array.
- Arrays are defined much like matrices, with a call to the **array()** command. Here is a 3D array (2×3×3):

```
> arr = array(1:18, dim=c(2,3,3))
> arr
, , 1
[,1] [,2] [,3]
[1,] 1 3 5
[2,] 2 4 6
, , 2
[,1] [,2] [,3]
[1,] 7 9 11
[2,] 8 10 12
, , 3
[,1] [,2] [,3]
[1,] 13 15 17
[2,] 14 16 18
```

R Programming; Factors

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

R Programming; Factors

A **factor** is a vector that represents **categorical** data

What is a **categorical variable**?

In a categorical variable, the value is limited (usually based on a particular finite group). While **continuous variable** could take any values.

Example: **gender** variable (it could be "male" of "female"), **cancer subtype** variable, **colour** variable and so on

In R, categorical variables are stored into a factor.

R Programming; Factors

Example: create a colour factor including 6 colours

```
# create a colour vector
colour_vector <- c('blue', 'red', 'green', 'white', 'black', 'yellow')
# convert the vector to factor
colour_factor <- factor(colour_vector)
colour_factor
```

```
[1] blue red green white black yellow
Levels: black blue green red white yellow
```

Levels: a vector of possible values taken by a factor varibale

R Programming; Factors

Each element of a factor comes from a pre-defined set of categories.
It can be:

- **ordinal** (ordered or ranked): small, medium, large.
- **nominal** (unordered): blue, yellow, green

Factors can be written or coded using any type (integer, character, logical).

```
# unordered 3-level factor with integers
x0 <- factor(c(1, 2, 3, 2))
x0
[1] 1 2 3 2 #
Levels: 1 2 3
```

```
table(x0)
x0
1 2 3
1 2 1
```

table(): building a table of the counts at each combination of factor levels.

R Programming; Factors

A **factor** is a vector that represents **categorical** data
Each element comes from a pre-defined set of categories.
It can be:

- **ordinal** (ordered or ranked): small, medium, large.
- **nominal** (unordered): blue, yellow, green

Factors can be written or coded using any mode (integer, text, logical).

```
# unordered 3-level factor with text (default order is alphanumeric)
x1 <- factor(c("large", "small", "medium", "small"))
Print(x1)
[1] large small medium small
Levels: large   medium   small
print((table(x1))
# large medium small
#    1       1       2
```

R Programming; Dataframes

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

R Programming; Data Frames

Creating a data frame

Filter

	name	score	attempts	qualify
1	Anastasia	12.5	1	yes
2	Dima	9.0	*NA*	no
3	Katherine	16.5	2	yes
4	James	12.0	*NA*	no
5	Emily	9.0	2	no
6	Michael	20.0	*NA*	yes
7	Matthew	14.5	1	yes
8	Laura	13.5	*NA*	no
9	Kevin	8.0	2	no
10	Jonas	19.0	1	yes

Showing 1 to 10 of 10 entries, 4 total columns

R Programming; Data Frames

Creating a data frame

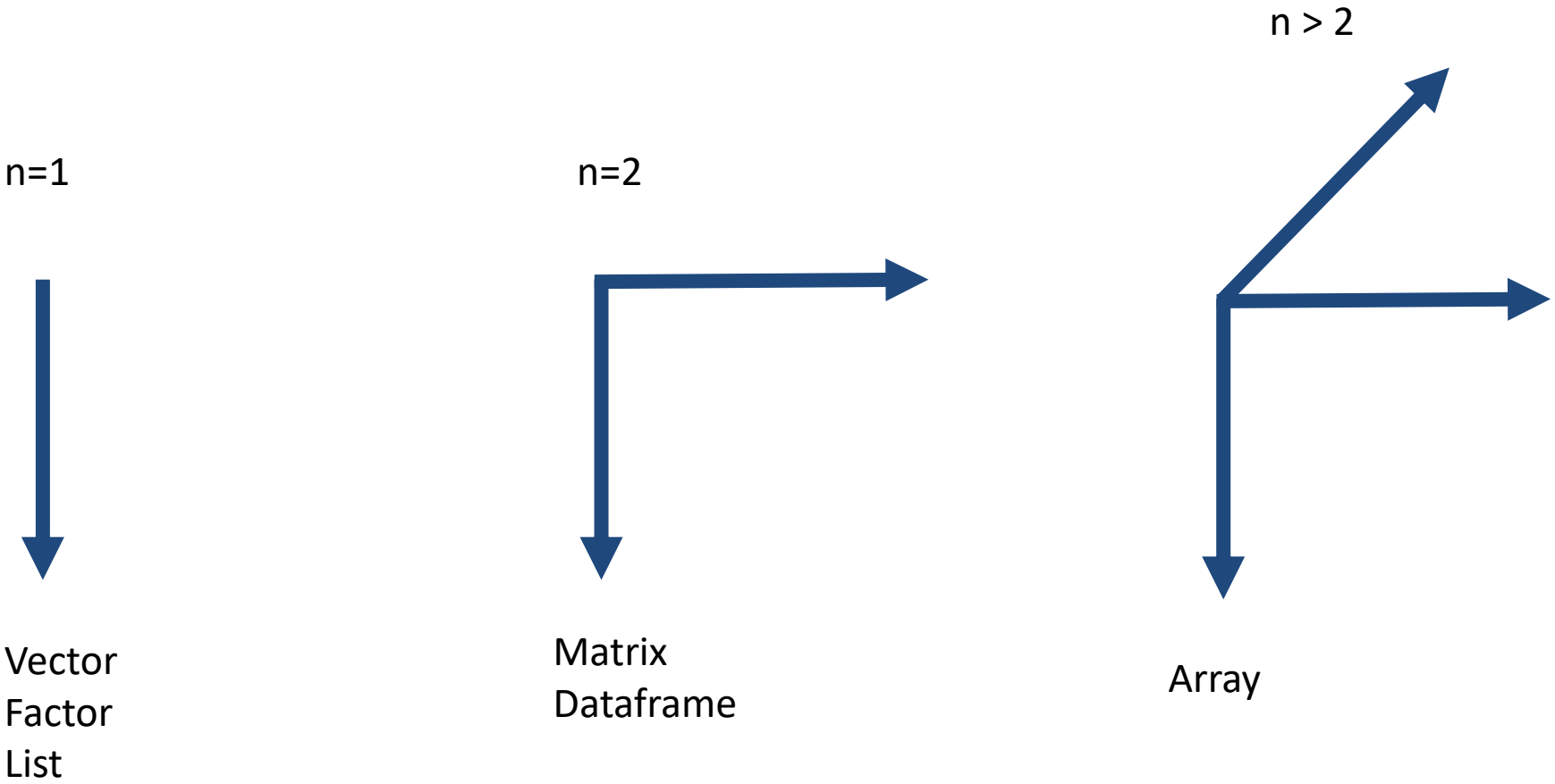
```
exam_data = data.frame( name = c('Anastasia', 'Dima', 'Katherine', 'James', 'Emily',
'Michael', 'Matthew', 'Laura', 'Kevin', 'Jonas'), score = c(12.5, 9, 16.5, 12, 9, 20,
14.5, 13.5, 8, 19), attempts = c(1, NA, 2, NA, 2, NA, 1, NA, 2, 1), qualify = c('yes',
'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes') )

print("Original dataframe:")
print(exam_data)
print("The number of NA values in attempts column:")
print(sum(is.na(exam_data$attempts)))
```

Filter

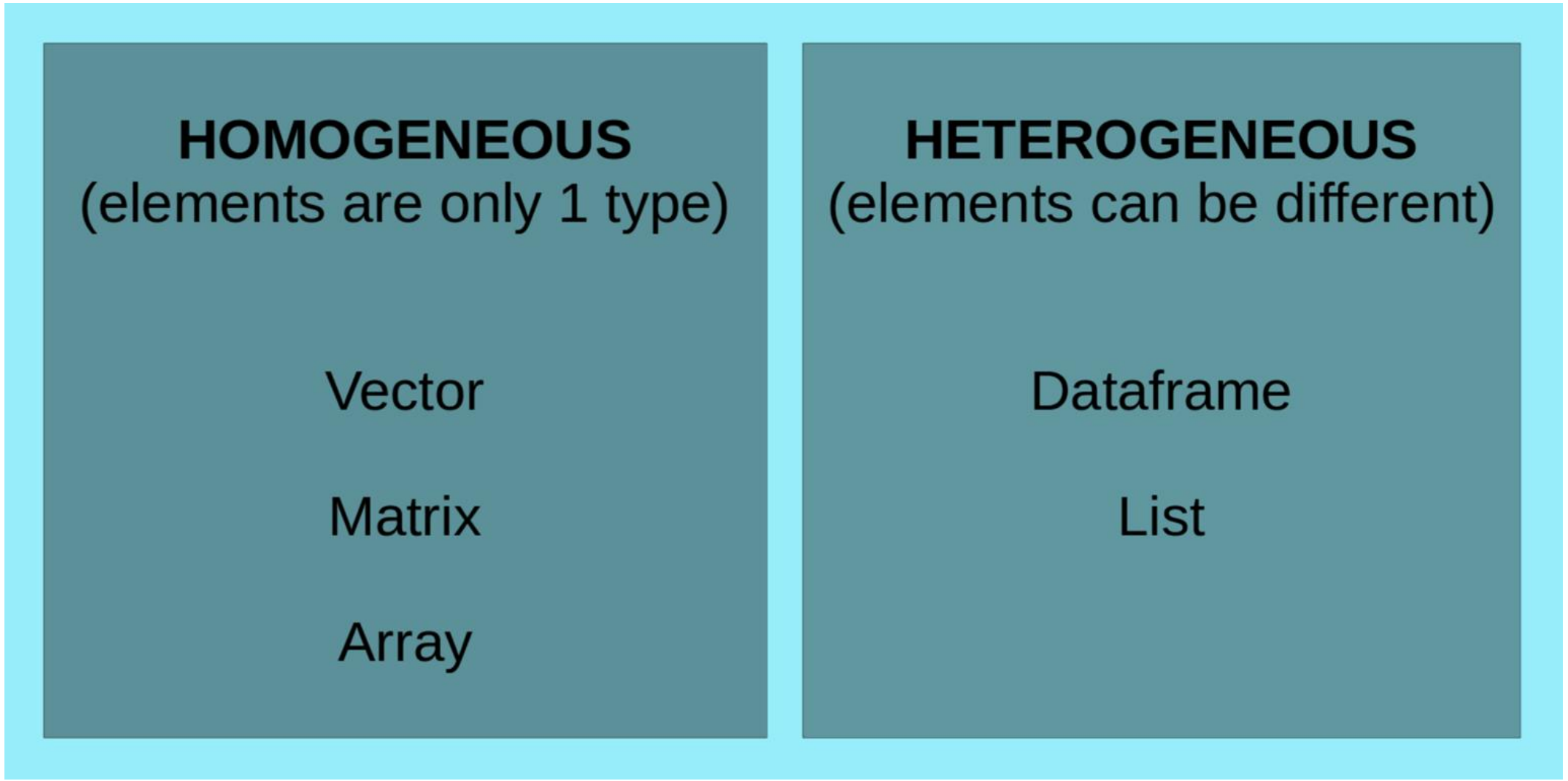
	name	score	attempts	qualify
1	Anastasia	12.5	1	yes
2	Dima	9.0	*NA*	no
3	Katherine	16.5	2	yes
4	James	12.0	*NA*	no
5	Emily	9.0	2	no
6	Michael	20.0	*NA*	yes
7	Matthew	14.5	1	yes
8	Laura	13.5	*NA*	no
9	Kevin	8.0	2	no
10	Jonas	19.0	1	yes

Showing 1 to 10 of 10 entries, 4 total columns

R Programming; Comparison



R Programming; Comparison



R Programming; R-Objects

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

Any Questions?