UNIVERSITY OF PORTSMOUTH

R for Data Analysis

Data Structures in R

(TB2 - Week 6)

Atefeh Khazaei

atefeh.khazaei@port.ac.uk

What we will learn this week?

- Data Structures in R
  - Vector
  - List
  - Matrix
  - Array
  - Data Frame
- Read Data

Vector

- VECTOR is the most basic object and default data structure in R
- Creating vectors using the following functions:
  - vector()
  - c()

```
V0 <- vector("integer", length = 5)
V0
class(V0)
```

```
0 · 0 · 0 · 0 · 0

'integer'
```

```
v1 <- c(1, 2, 3, 4, 5)
v1
is.vector(v1)
class(v1)
```

```
1 · 2 · 3 · 4 · 5

TRUE

'numeric'
```

Vector (cont.)

```
v2 <- c("a", "b", "c")
v2
is.vector(v2)
class(v2)
```

'a' · 'b' · 'c'

TRUE

'character'

```
v3 <- c(TRUE, TRUE, FALSE, FALSE, TRUE)
v3
is.vector(v3)
class(v3)
```

TRUE · TRUE · FALSE · FALSE · TRUE

TRUE

'logical'

```
v4 <- c(TRUE, "a", 2.5, 3)
v4
is.vector(v4)
class(v4)
```

'TRUE' · 'a' · '2.5' · '3'

TRUE

'character'

```
v5 <- c(v3,v4)
v5
is.vector(v5)
class(v5)
```

'TRUE' · 'TRUE' · 'FALSE' · 'FALSE' · 'TRUE' · 'TRUE' · 'a' · '2.5' · '3'

TRUE

'character'

Vector (cont.)

- **as.** is a command to convert different types together.
  - as.integer
  - as.numeric
- Do more example:
  - Try to convert logical class to integer and character.
  - Try to convert character class integer.

```
v1 <- c(1, 2, 3, 4, 5)
v1
is.vector(v1)
class(v1)
```

```
1 · 2 · 3 · 4 · 5

TRUE

'numeric'
```

```
v1 = as.character(v1)
v1
class(v1)
```

```
'1' · '2' · '3' · '4' · '5'

'character'
```

List

- LIST is a generic vector containing objects.

```
n = c(2,3,5)
s = c("aa", "bb", "cc", "dd")
b = c(TRUE, FALSE)

lst = list(n,s,b,3)
lst
```

1. 2 · 3 · 5
2. 'aa' · 'bb' · 'cc' · 'dd'
3. TRUE · FALSE
4. 3

```
lst[2]
```

1. 'aa' · 'bb' · 'cc' · 'dd'

```
lst[c(2,4)]
```

1. 'aa' · 'bb' · 'cc' · 'dd'
2. 3

```
lst[c(2,4)][[1]]
```

'aa' · 'bb' · 'cc' · 'dd'

```
lst[c(2,4)][[1]][1]
```

'aa'

Matrix

- MATRIX is a collection of data elements arranged in a two-dimensional rectangular layout.
- Create a matrix using **matrix()** function in R:

```
m1 <- matrix(c(T, T, F, F, T, F), nrow = 2)
m1
```

A matrix: 2 × 3 of type lgl

TRUE	FALSE	TRUE
TRUE	FALSE	FALSE

```
m2 <- matrix(c("a", "b",
               "c", "d"),
             ncol = 2,
             byrow = T)
m2
```

A matrix: 2 × 2 of type chr

a	b
c	d

Matrix (cont.)

```
m3 = matrix(c(1,2,3,4,5,6), nrow=3, ncol=2)
m3
```

A
matrix:
3 × 2
of type
dbl

1	4
2	5
3	6

```
m3[2,1]
```

2

```
m3[,2]
```

4 · 5 · 6

```
m3[3,]
```

3 · 6

Matrix (cont.)

- The columns of two matrices having the same number of rows can be combined into a larger matrix.

```
m3 = matrix(c(1,2,3,4,5,6), nrow=3, ncol=2)
m4 = matrix(c(7,8,9), nrow=3, ncol=1)

m5 = cbind(m3,m4)
m5
```

A matrix:

3 × 3 of

type dbl

1	4	7
2	5	8
3	6	9

Matrix (cont.)

- You can deconstruct a matrix by applying the c function, it will combine all columns into one.

```
m3 = matrix(c(1,2,3,4,5,6), nrow=3, ncol=2)
m3
```

A
matrix:
3 × 2
of type
dbl

1	4
2	5
3	6

```
c(m3)
```

1 · 2 · 3 · 4 · 5 · 6

Array

- In matrix we only have two dimensions, but array is more flexible and you can have more than two dimensions.

```
# Give data, then dimensions (rows, columns, tables)
a1 <- array(c( 1:24), c(4, 3, 2))
a1
```

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

```
> a1 <- array(c( 1:24), c(4, 3, 2))
> a1
, , 1

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

, , 2

     [,1] [,2] [,3]
[1,]   13   17   21
[2,]   14   18   22
[3,]   15   19   23
[4,]   16   20   24
```

Data Frame

- Data Frame is more flexible than matrix and array.
- A universal container for large data sets.

```r
# Data Frame Can combine vectors of the same length

vNumeric   <- c(1, 2, 3)
vCharacter <- c("a", "b", "c")
vLogical   <- c(T, F, T)

df1 <- cbind(vNumeric, vCharacter, vLogical)
df1  # Coerces all values to most basic data type

df2 <- as.data.frame(cbind(vNumeric, vCharacter, vLogical))
df2  # Makes a data frame with three different data types
```

Character in this example

A matrix: 3 × 3 of type chr

vNumeric	vCharacter	vLogical
1	a	TRUE
2	b	FALSE
3	c	TRUE

A data.frame: 3 × 3

vNumeric	vCharacter	vLogical
<fct>	<fct>	<fct>
1	a	TRUE
2	b	FALSE
3	c	TRUE

Data Frame (cont.)

- Many data input functions of R like, read.table(), read.csv(), read.delim(), read.fwf() also read data into a data frame.
- Use either [, [[or $ to access a column values of data frame.

A data.frame: 3 × 3

vNumeric <fct>	vCharacter <fct>	vLogical <fct>
1	a	TRUE
2	b	FALSE
3	c	TRUE

```
df2["vCharacter"]
```

A data.frame: 3 × 1

vCharacter <fct>
a
b
c

```
df2[["vCharacter"]]
```

a · b · c

▼ **Levels**:

'a' · 'b' · 'c'

```
df2[[3]]
```

TRUE · FALSE · TRUE

▼ **Levels**:

'FALSE' · 'TRUE'

Data Frame (cont.)

```
head(df2,n=2)
```

A data.frame: 2 × 3

	vNumeric	vCharacter	vLogical
	<fct>	<fct>	<fct>
1	1	a	TRUE
2	2	b	FALSE

```
df2[,2:3]
```

A data.frame: 3 × 2

vCharacter	vLogical
<fct>	<fct>
a	TRUE
b	FALSE
c	TRUE

```
df2[df2$vNumeric == 2,]
```

A data.frame: 1 × 3

	vNumeric	vCharacter	vLogical
	<fct>	<fct>	<fct>
2	2	b	FALSE

Data Frame (cont.)

```r
vNumeric   <- c(1, 2, 3)
vCharacter <- c("a", "b", "c")
vLogical   <- c(T, F, T)

df2 <- as.data.frame(cbind(vNumeric, vCharacter, vLogical), stringsAsFactors=FALSE)
df2  # Makes a data frame with three different data types
```

A data.frame: 3 × 3

vNumeric	vCharacter	vLogical
<chr>	<chr>	<chr>
1	a	TRUE
2	b	FALSE
3	c	TRUE

```r
df2[df2$vNumeric == 2, "vNumeric"] <- 5
df2
```

A data.frame: 3 × 3

vNumeric	vCharacter	vLogical
<chr>	<chr>	<chr>
1	a	TRUE
5	b	FALSE
3	c	TRUE

Read Data

```
df3=read.csv("http://vincentarelbundock.github.io/Rdatasets/csv/datasets/USArrests.csv", stringsAsFactors = FALSE)
nrow(df3)
ncol(df3)
head(df3)
```

50

5

A data.frame: 6 × 5

	X	Murder	Assault	UrbanPop	Rape
	<chr>	<dbl>	<int>	<int>	<dbl>
1	Alabama	13.2	236	58	21.2
2	Alaska	10.0	263	48	44.5
3	Arizona	8.1	294	80	31.0
4	Arkansas	8.8	190	50	19.5
5	California	9.0	276	91	40.6
6	Colorado	7.9	204	78	38.7

Read Data (cont.)

```
dim(df3)
```

```
50 · 5
```

```
colnames(df3)
```

```
'X' · 'Murder' · 'Assault' · 'UrbanPop' · 'Rape'
```

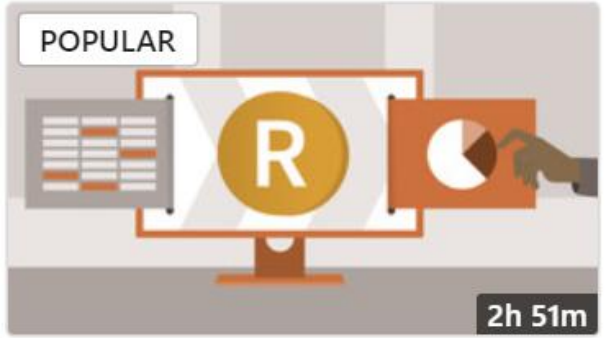
```
summary(df3)
```

```
     X                  Murder          Assault         UrbanPop    
 Length:50          Min.   : 0.800   Min.   : 45.0   Min.   :32.00  
 Class :character   1st Qu.: 4.075   1st Qu.:109.0   1st Qu.:54.50  
 Mode  :character   Median : 7.250   Median :159.0   Median :66.00  
                    Mean   : 7.788   Mean   :170.8   Mean   :65.54  
                    3rd Qu.:11.250   3rd Qu.:249.0   3rd Qu.:77.75  
                    Max.   :17.400   Max.   :337.0   Max.   :91.00  
      Rape       
 Min.   : 7.30  
 1st Qu.:15.07  
 Median :20.10  
 Mean   :21.23  
 3rd Qu.:26.18  
 Max.   :46.00  
```

References & More Resources

- References:
  - Learning R:

    https://www.linkedin.com/learning/learning-r-2/



  - R Programming in Data Science: Setup and Start

    https://www.linkedin.com/learning/r-programming-in-data-science-setup-and-start/

  - To use LinkedinLearning, you can log in with your university account:

    https://myport.port.ac.uk/study-skills/linkedin-learning

Practical Session

- Try these slides' examples to understand data structures better.