Introduction to Data Science (11372 & G 11516)

Semester 1 2021

UNIVERSITY OF CANBERRA

# INTRODUCTION TO DATA SCIENCE

## Lecture 4

Dr. Ibrahim Radwan

DISTINCTIVE BY DESIGN

# OUTLINE

UNIVERSITY OF CANBERRA

- Data Structures
  - Factors
  - Arrays and Matrices
  - Lists
  - Data Frames

# DATA STRUCTURES

UNIVERSITY OF CANBERRA

## Similar type of items

### Homogeneous data structures

| | |
|---|---|
| Atomic vector | 1-D |
| Matrix | 2-D |
| Array | n-D |

## Dissimilar type of items

### Heterogeneous data structures

| | |
|---|---|
| List | 1-D |
| Data frame | 2-D |

# FACTOR

UNIVERSITY OF CANBERRA

- Special case of vector, used to store nominal/categorical data
- A factor can be thought as an integer vector, where each integer has a label
  - It is more efficient to have a variable with “Female” and “Male” values than a variable with 1 and 2 values
  - Integers are more efficient when comparing values or when searching for a value than using characters
  - However, integers are not self describing and characters are
  - Factor is the solution that integrates both
- Creating a factor
  - `x <- factor(c("yes", "yes", "no", "yes", "no"))`

# FACTOR (2)

UNIVERSITY OF CANBERRA

Code Example

```r
# Factor example

# Create character vector for types of the properties
houses.types <- c("House", "Unit", "House", "Unit")
houses.types
str(houses.types)
# Create integral vector for types of the properties
houses.types <- c(1L, 2L, 1L, 2L)
houses.types
str(houses.types)

# create a factor
houses.types <- factor(c("House", "Unit", "House", "Unit"))
houses.types
# underlying representation of factor
unclass(houses.types)
str(houses.types)
# Searching using characters
houses.types[houses.types == "House"] # not effecient
# Searching using integer
houses.types[as.numeric(houses.types) == 1] # effecient

#
employees.gender <- rep(c("female", "male", "male", "female", "male"), 10)
employees.gender
employees <- factor(employees.gender)
str(employees)

# order of levels
x <- factor(c("yes", "no", "yes", "no"))
x
y <- factor(c("yes", "no", "yes", "no"), levels= c("yes", "no"))
y
```

# ARRAYS AND MATRICES

UNIVERSITY OF CANBERRA

- Arrays store data elements in several dimensions. Matrices are special cases of arrays with only two dimensions. Arrays and Matrices in R are nothing more than vectors with a attribute that is the dimension.
- Creating a matrix:
  - `m <- matrix(c(45, 23, 66, 77, 33, 44, 56, 12, 78, 23), 2, 5)`

Elements of the matrix

nrows

ncols

- Creating an array:
  - `m <- array(c(45, 23, 66, 77, 33, 44, 56, 12), c(2, 2, 2))`

Elements of the array

dimensions

# ARRAYS AND MATRICES (2)

UNIVERSITY OF CANBERRA

Code Example

```r
# arrays
my.array <- array(1:24, dim=c(3,4,2))
my.array
# Q: how are the items ordered?
# A: It is ordered column-wise

# change the dimensions of a vector in R
my.vector <- 1:24
dim(my.vector) <- c(3,4,2)

# check that the contents of two objects are equal using identical function
identical(my.array, my.vector)

# matrices
my.matrix <- matrix(1:24, nrow= 4, ncol= 6)
my.matrix
```

# ARRAYS AND MATRICES (3)

UNIVERSITY OF CANBERRA

Code example for sub-setting

```r
# Subsetting arrays and matrices
m <- matrix(c(45, 23, 66, 77, 33, 44, 56, 12, 78, 23, 38, 17), nrow= 3, ncol=4)
m
# extract the element of the second row and third column
m[2, 3]
# extract all elements of the second row
m[2,]
# extract all elements of the third column
m[,3]
# extract all element of the second row except the the third element of this row
m[2, -3]
# extract the second and fourth elements of the second row
m[2, c(2, 4)]
# extract all elements of the second row except the second and fourth elements
m[2, -c(2, 4)]

# all of the above are either column or row, so the results were vectors
# The results can be a sub-matrix
a <- m[c(1,2), c(2, 4)]
a
class(a)
# If you want the result of subsetting to be a matrix even for row and column
b <- m[1, ,drop= FALSE] # result in matrix
b
class(b)
```

# ARRAYS AND MATRICES (4)

UNIVERSITY OF CANBERRA

- You can use *cbind()* and *rbind()* *functions* to join two or more vectors together or matrices, by columns or by rows, respectively. The following examples illustrates this:

Code example

```
1  # rbind and cbind example
2  m1 <- matrix(c(45, 23, 66, 77, 33, 44, 56, 12, 78, 23, 38, 17), nrow=3, ncol=4)
3  m1
4  res <- cbind(c(4, 76, 12), m1[, 4])
5  res
6
7  res <- rbind(c(4, 76, 12, 14), m1[3, ])
8  res
9
10 m2 <- matrix(rep(10, 16), nrow= 4, ncol= 4)
11 m2
12
13 m3 <- rbind(m1[1, ], m2[3, ])
14 m3
```

# ARRAYS AND MATRICES (5)

UNIVERSITY OF CANBERRA

- You can also give names to the columns and rows of matrices, using the functions
- *colnames()* and *rownames()*. This facilitates memorizing the data positions.

Code example

```
1  # using rownames() and colnames() functions
2  results <- matrix(c(10, 30, 40, 50, 43, 56, 21, 30), 2, 4, byrow= TRUE)
3
4  # names of the columns
5  colnames(results) <- c("1st_qrt", "2nd_qrt", "3rd_qrt", "4th_qrt")
6
7  # names of the rows
8  rownames(results) <- c("store1", "store2")
9
10 # show the matrix
11 results
12
13 # indexing using the names of the columns/rows
14 results["store1", ]
15 results["store2", c("1st_qrt", "4th_qrt")]
```

You can also use *dim(matrix)* to get the dimensions of arrays or matrices

# LISTS

UNIVERSITY OF CANBERRA

- A list is a data structure that holds collection of possibly unrelated (or heterogeneous) objects in order under one name.
  - Think of a vector, when you would like to combine the name, age, salary of an employee in one vector, is it possible?
  - No, most likely all of the numeric values (e.g. age and salary) will be converted to character to fulfill the rule that the items of a vector have to be from the same data type
  - However, with the lists, this is possible:
    - `> employee <- list(name='Jack', age=25L, salary=5210.43)`

Different type of objects are concatenated together

- Combining two lists will results in a list

# LISTS (2)

UNIVERSITY OF CANBERRA

Code Example

```
# vectors
# Character
houses.addresses <- c("7 George st", "18/5 Irwan cresent", "8 Morad close", "1/2 London Circuit")
# Numeric
houses.area <- c(420.5, 220.15, 750.4, 120.5)
# Integer
houses.bedrooms <- c(4L, 3L, 5L, 2L)
# Logical
houses.has.garden <- c(TRUE, FALSE, TRUE, FALSE)

# Combine the features of each proprty
# using vector
houses.sample <- c(address= houses.addresses[1], area=houses.area[1], bedrooms= houses.bedrooms[1],
                   has_garden=houses.has.garden[1])
print(houses.sample)
mode(houses.sample)
houses.sample[1]
houses.sample[2]
houses.sample['bedrooms']
str(houses.sample)
# using list
houses.sample <- list(address= houses.addresses[1], area=houses.area[1], bedrooms= houses.bedrooms[1],
                      has_garden=houses.has.garden[1])
print(houses.sample)
mode(houses.sample)
houses.sample[1]
houses.sample[2]
houses.sample['bedrooms']
houses.sample$bedrooms
mode(houses.sample$bedrooms)
str(houses.sample)
```

# DATA FRAMES

UNIVERSITY OF CANBERRA

- A data frame is a series of records represented by rows (observations), where each row contains values in several fields/columns (variables).
- They are like matrices in structure as they are also bi-dimensional.
  - All the operations, we have used for matrices can be applied on data frames as well
    - such as rbind(), cbind(), dim(), ...
- However, contrary to matrices, data frames may include data of a different type in each column.
- So, a data frame is a special type of lists, where a list represents only one row (in-order) and a data frame can be one row or more.

# DATA FRAMES (2)

UNIVERSITY OF CANBERRA

ID, Name, Age

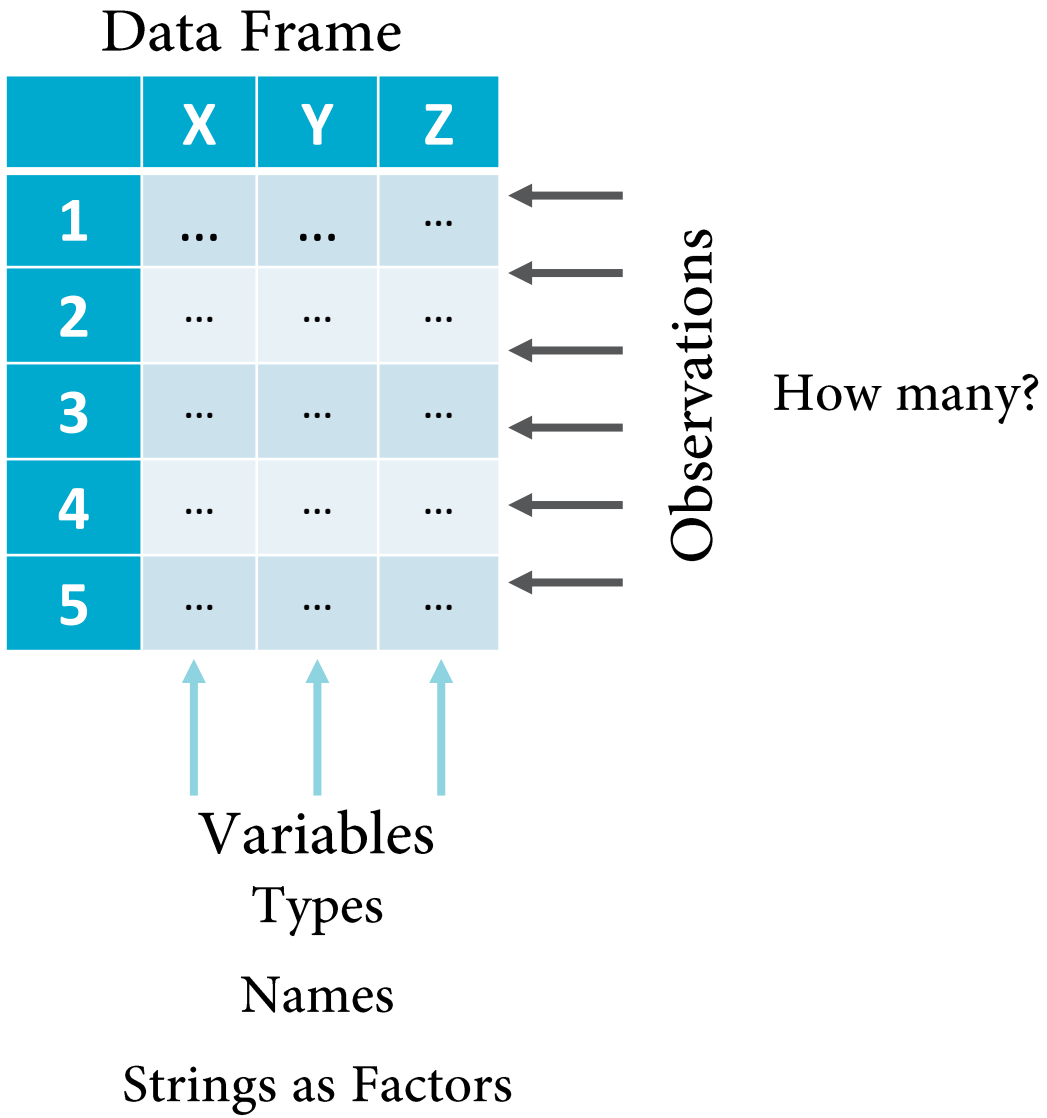
23424, Ana, 45

11234, Charles, 23

77654, Susanne, 76

data.csv

How to read this using the base functions in R?



# DATA FRAMES (3)

UNIVERSITY OF CANBERRA

- Create data frames from vectors:

```
# create dataframe from vectors
employee <- c('John Doe','Peter Gynn','Jolie Hope')
salary <- c(21000, 23400, 26800)
startdate <- as.Date(c('2010-11-1','2008-3-25','2007-3-14'))
employ.data <- data.frame(employee, salary, startdate)
```

- Check structure of data frame

```
str(employ.data)
```

- Characters by the default are set as Factors, however this can be changed using:

```
employ.data <- data.frame(employee, salary, startdate,
stringsAsFactors = FALSE)
```

- Full example in the next slide

# DATA FRAMES (4)

UNIVERSITY OF CANBERRA

Code Example

```r
# create dataframe from vectors
employee <- c('John Doe','Peter Gynn','Jolie Hope')
salary <- c(21000, 23400, 26800)
startdate <- as.Date(c('2010-11-1','2008-3-25','2007-3-14'))

employ.data <- data.frame(employee, salary, startdate)
# check structure of data frame
str(employ.data)

# keep characters as characters
employ.data <- data.frame(employee, salary, startdate, stringsAsFactors = FALSE)
str(employ.data)

# using the accessor $ to access the variables of the data frame
employ.data$employee
employ.data$salary
employ.data$startdate[1]

# what is the difference between the following expressions
aa <- employ.data["salary"]
bb <- employ.data[["salary"]]
aa
class(aa)
bb
class(bb)
```

# DATA FRAMES (5)

UNIVERSITY OF CANBERRA

- Sub-setting data frame using [] or accessor operator, $:

```
my.dataset <- data.frame(site=c("A", "B", "A", "A", "B"),
        season=c("Winter", "Summer", "Summer", "Spring", "Fall"),
        pH=c(7.4,6.3,8.6,7.2,8.9), stringsAsFactors = FALSE)
my.dataset[3, 2]
my.dataset$pH
```

- Sub-setting based on conditions:

```
my.dataset[my.dataset$pH > 7, ]
my.dataset[my.dataset$pH > 7, "site"]
my.dataset[my.dataset$season == "Summer", c("site", "pH")]
```

- Is it possible to refer to the columns directly?
  - Like we say, my.dataset[pH > 7, ]
  - No this is not possible unless we attach. Let us discuss the full example in the next slide.

# DATA FRAMES (6)

UNIVERSITY OF CANBERRA

Code Example

```r
# create and manipulate data frame
my.dataset <- data.frame(site=c("A", "B", "A", "A", "B"),
                         season=c("Winter", "Summer", "Summer", "Spring", "Fall"),
                         pH=c(7.4,6.3,8.6,7.2,8.9), stringsAsFactors = FALSE)
my.dataset
my.dataset[3, 2]
my.dataset$pH
# subsetting with conditions
my.dataset[my.dataset$pH > 7, ]
my.dataset[my.dataset$pH > 7, "site"]
my.dataset[my.dataset$season == "Summer", c("site", "pH")]
# Is it possible to refer to the columns directly?
my.dataset[pH > 7, ]
# This is only possible if you `attach` the dataset
attach(my.dataset)
my.dataset[pH > 7, ]
season
my.dataset[site=='B', ]
# To go back, use `detach`
detach(my.dataset)
season
# it is much safer to use `subset`
subset(my.dataset, pH > 8)
subset(my.dataset, season=="Summer", select=c(season,pH))
# change values in the dataframe, e.g., sum 1 to the pH values of all summer rows
my.dataset[my.dataset$season == "Summer", 'pH'] <-
  my.dataset[my.dataset$season == "Summer", 'pH'] + 1

subset(my.dataset, season=="Summer", select=c(season,pH))
```

# DATA FRAMES (7)

UNIVERSITY OF CANBERRA

- Some other operations on R:

```
#1- add new column to the data frame, must be the same number of rows
my.dataset$N03 <- c(234.5, 256.6, 654.1, 356.7, 776.4)
#2- check number of rows
nrow(my.dataset)
#3- check number of columns
ncol(my.dataset)
#4- check dimension
dim(my.dataset)
#5- edit the existing dataset
my.dataset <- edit(my.dataset)
#6- create new dataset and open it in the edit mode
new.data <- edit(data.frame())
#7- check names of the columns
names(my.dataset)
#8- change names of the columns
names(my.dataset) <- c("area", "season", "pH", "N03")
```

# KEY TAKEAWAYS

UNIVERSITY OF CANBERRA

- Matrices and Arrays are tabular forms that hold data from the same types
- Lists are like vectors, but for heterogeneous data types
- Data frames are the most commonly used data structure in R
- Data frames combine features of lists and matrices together

# RECOMMENDED READING

UNIVERSITY OF CANBERRA

- You are recommended to read section 3.5 from the following online book:
  - https://rafalab.github.io/dsbook/r-basics.html#data-frames
- Also, you may check some of the following links for some tutorial about data frames:
  - https://www.datamentor.io/r-programming/data-frame/
  - http://www.r-tutor.com/r-introduction/data-frame
  - https://www.tutorialspoint.com/r/r_data_frames.htm

# ANNOUNCEMENTS

UNIVERSITY OF CANBERRA

- Unit readiness test is available, due on Sunday 7th of March
- The solutions of the weekly lab exercises will be available every subsequent Monday.
- The census date is due this Friday, 5th of March