Practical No: 2-3

Aim: Data visualization using R Programming (Consider different input files like csv, excel, JSON etc.)

Theory:

In R, you can read data from files outside of the R environment. One may also write data to files that the operating system can store and further access. There is a wide range of file formats, including CSV, Excel, binary, and XML, etc., R can read and write from.
While many organizations store data in databases and storage options such as AWS, Azure, and GCP, Microsoft Excel spreadsheets continue to be widely used for storing smaller datasets.

Excel's data science functionality is more limited than R's, so it's useful to be able to import data from spreadsheets to R.

We may work with structured data from spreadsheets, take advantage of R's capabilities for data analysis and manipulation, and incorporate Excel data into other R processes and packages by reading Excel files in R. The readxl package offers a simple and effective method for reading Excel files into R as data frames for additional processing and analysis.

Reading Data from Excel Files in R

These files are used to store data in a tabular form and are commonly employed in data analysis and manipulation tasks. Must have worked with structured data from spreadsheets, can now take advantage of R's capabilities for data analysis and manipulation, and incorporate Excel data into other R processes and packages by reading Excel files in R.

R provides several packages like readxl, xlsx, and openxlsx to read or import Excel files into R DataFrame. These packages provide several methods with different arguments which help us read Excel files effectively.

Two different techniques to read or import an Excel file in R.
Method 1: Using read_excel() from readxl

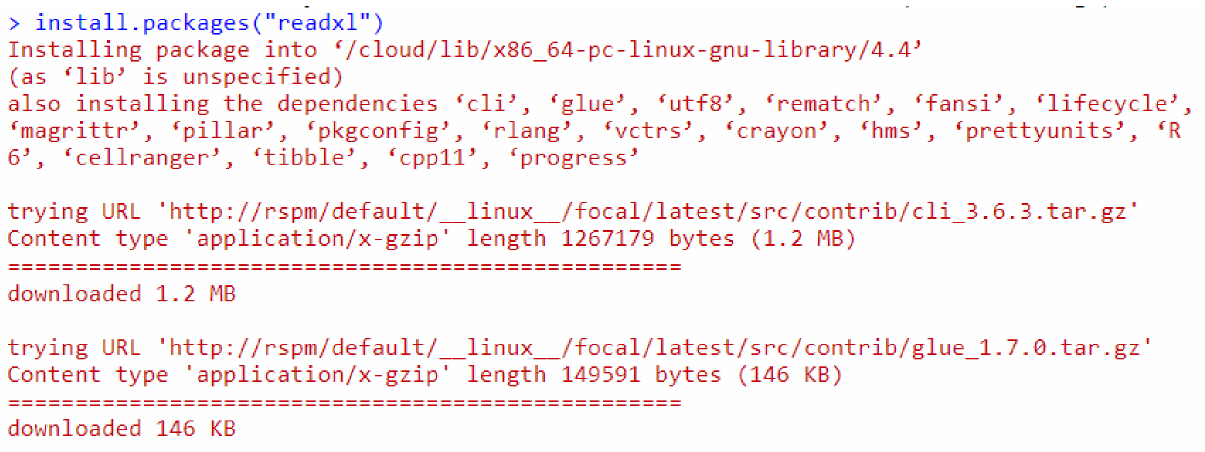
Method 2: Using read.xlsx() from xlsx

Approach
- Import module
- Pass the path of the file to the required function
- Read file
- Display content

Step 1: Installing Necessary Packages

```
# Install the necessary packages
install.packages("readxl")
install.packages("writexl")
```

The "install.packages()" function is used to install packages in R. the "readxl" package is being installed. This package provides functions for reading Excel files into R. See below output in the console, signaling successful installation.

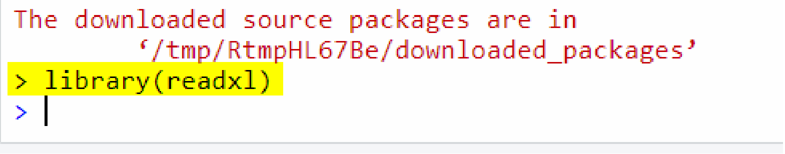


```
> install.packages("readxl")
Installing package into '/cloud/lib/x86_64-pc-linux-gnu-library/4.4'
(as 'lib' is unspecified)
also installing the dependencies 'cli', 'glue', 'utf8', 'rematch', 'fansi', 'lifecycle',
'magrittr', 'pillar', 'pkgconfig', 'rlang', 'vctrs', 'crayon', 'hms', 'prettyunits', 'R
6', 'cellranger', 'tibble', 'cpp11', 'progress'

trying URL 'http://rspm/default/__linux__/focal/latest/src/contrib/cli_3.6.3.tar.gz'
Content type 'application/x-gzip' length 1267179 bytes (1.2 MB)
==================================================
downloaded 1.2 MB

trying URL 'http://rspm/default/__linux__/focal/latest/src/contrib/glue_1.7.0.tar.gz'
Content type 'application/x-gzip' length 149591 bytes (146 KB)
==================================================
downloaded 146 KB
```

Step 2: Loading Packages

# Load the necessary packages
library(readxl)
library(writexl)

The code loads the readxl package in R. The library() function is used to load packages in R, which are collections of functions and data sets that extend the functionality of R. The readxl package provides functions for reading Excel files into R. By loading this package, the user can access these functions and use them in their R code.



```
The downloaded source packages are in
        '/tmp/RtmpHL67Be/downloaded_packages'
> library(readxl)
> 
```

Step 3: Reading an Excel File

Use the function read_excel() from the 'readxl' package. This function requires as an argument the path to the Excel file.

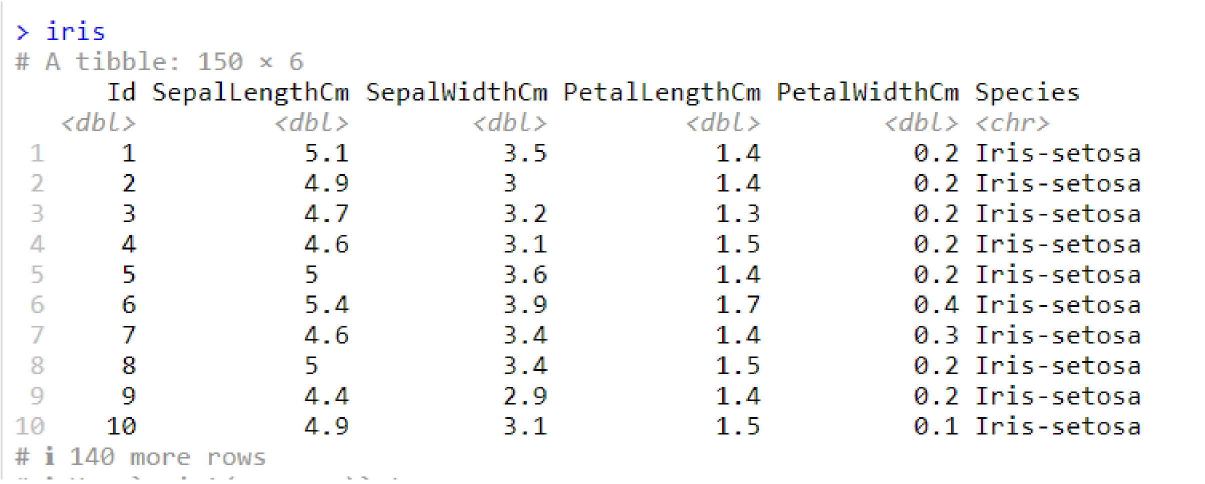
# Read an Excel file
data <- read_excel("path/to/your/file.xlsx")

iris <- read_xlsx("sample-dataset 2-3.xlsx", sheet = "iris")

OR

iris2 <- read_xlsx("sample-dataset 2-3.xlsx", sheet = 1)

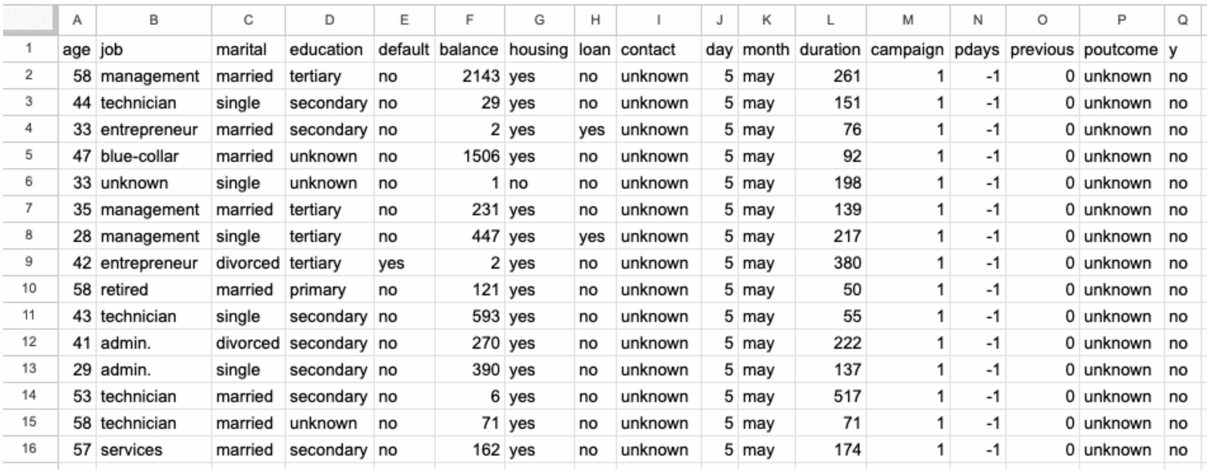
The code reads an Excel file named "sample.xlsx" and extracts the data from the sheet named "iris". The data is then stored in a data frame named "iris". The "<" symbol is an HTML entity that represents the less than sign "<". In R, the less than sign is used for assignment, so this code assigns the data from the Excel sheet to the "iris" data frame.



```
> iris
# A tibble: 150 × 6
      Id SepalLengthCm SepalWidthCm PetalLengthCm PetalWidthCm Species
   <dbl>         <dbl>        <dbl>         <dbl>        <dbl> <chr>
 1     1           5.1          3.5           1.4          0.2 Iris-setosa
 2     2           4.9          3             1.4          0.2 Iris-setosa
 3     3           4.7          3.2           1.3          0.2 Iris-setosa
 4     4           4.6          3.1           1.5          0.2 Iris-setosa
 5     5           5            3.6           1.4          0.2 Iris-setosa
 6     6           5.4          3.9           1.7          0.4 Iris-setosa
 7     7           4.6          3.4           1.4          0.3 Iris-setosa
 8     8           5            3.4           1.5          0.2 Iris-setosa
 9     9           4.4          2.9           1.4          0.2 Iris-setosa
10    10           4.9          3.1           1.5          0.1 Iris-setosa
# ℹ 140 more rows
```

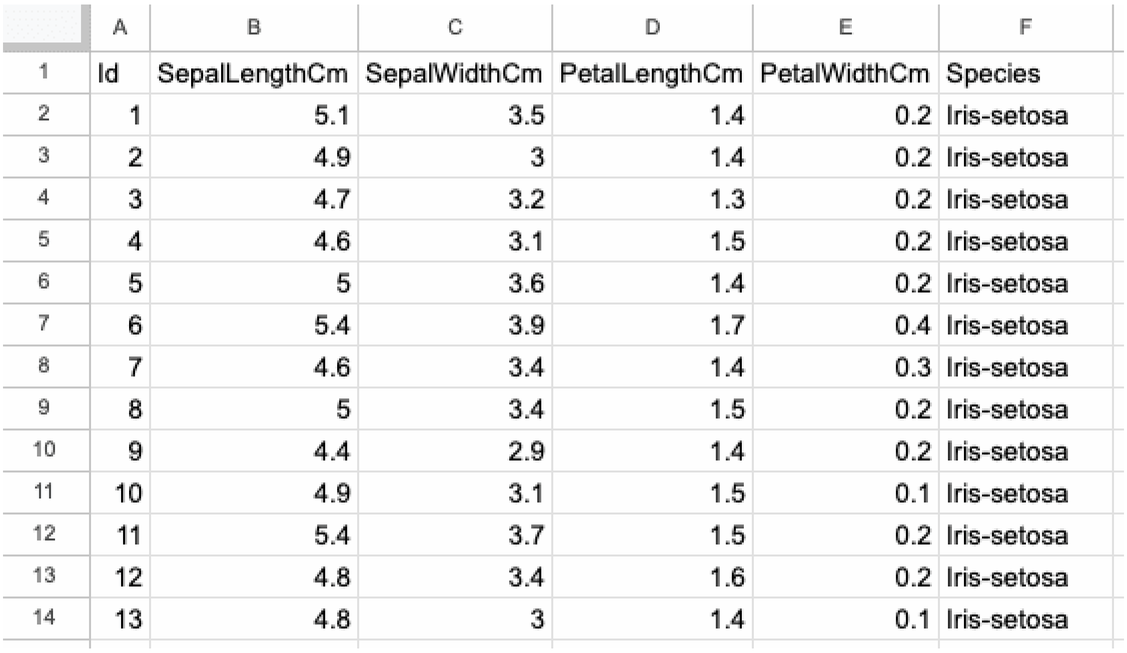
About the dataset used:

The dataset read into R is a small one with only two sheets to demonstrate how to specify which sheet to read. The first sheet is a bank marketing dataset with 45,211 rows and 17 columns. The screenshot below is from the excel file “sample-dataset 2-3.xlsx” and sheet name “bank-full”.

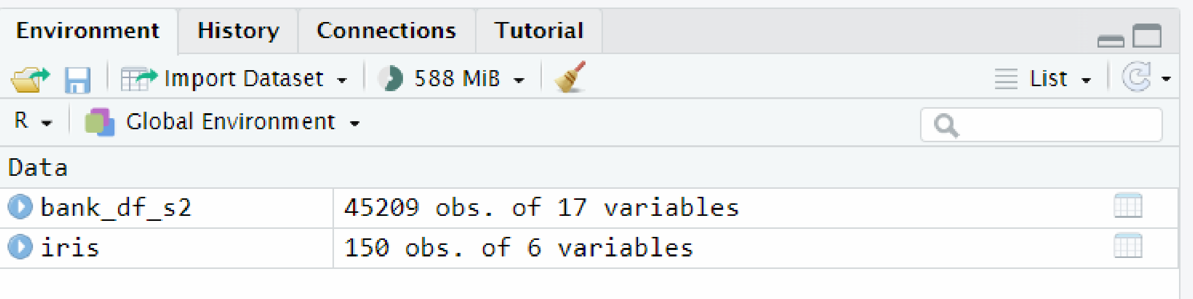


	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
1	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	pdays	previous	poutcome	y
2	58	management	married	tertiary	no	2143	yes	no	unknown	5	may	261	1	-1	0	unknown	no
3	44	technician	single	secondary	no	29	yes	no	unknown	5	may	151	1	-1	0	unknown	no
4	33	entrepreneur	married	secondary	no	2	yes	yes	unknown	5	may	76	1	-1	0	unknown	no
5	47	blue-collar	married	unknown	no	1506	yes	no	unknown	5	may	92	1	-1	0	unknown	no
6	33	unknown	single	unknown	no	1	no	no	unknown	5	may	198	1	-1	0	unknown	no
7	35	management	married	tertiary	no	231	yes	no	unknown	5	may	139	1	-1	0	unknown	no
8	28	management	single	tertiary	no	447	yes	yes	unknown	5	may	217	1	-1	0	unknown	no
9	42	entrepreneur	divorced	tertiary	yes	2	yes	no	unknown	5	may	380	1	-1	0	unknown	no
10	58	retired	married	primary	no	121	yes	no	unknown	5	may	50	1	-1	0	unknown	no
11	43	technician	single	secondary	no	593	yes	no	unknown	5	may	55	1	-1	0	unknown	no
12	41	admin.	divorced	secondary	no	270	yes	no	unknown	5	may	222	1	-1	0	unknown	no
13	29	admin.	single	secondary	no	390	yes	no	unknown	5	may	137	1	-1	0	unknown	no
14	53	technician	married	secondary	no	6	yes	no	unknown	5	may	517	1	-1	0	unknown	no
15	58	technician	married	unknown	no	71	yes	no	unknown	5	may	71	1	-1	0	unknown	no
16	57	services	married	secondary	no	162	yes	no	unknown	5	may	174	1	-1	0	unknown	no

The second sheet is the Iris dataset with 150 rows and 6 columns and contains information about Iris flower types, such as their sepal and petal lengths and widths. The screenshot below is from the same excel file, “sample.xlsx” and sheet name “iris”.



	A	B	C	D	E	F
1	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
2	1	5.1	3.5	1.4	0.2	Iris-setosa
3	2	4.9	3	1.4	0.2	Iris-setosa
4	3	4.7	3.2	1.3	0.2	Iris-setosa
5	4	4.6	3.1	1.5	0.2	Iris-setosa
6	5	5	3.6	1.4	0.2	Iris-setosa
7	6	5.4	3.9	1.7	0.4	Iris-setosa
8	7	4.6	3.4	1.4	0.3	Iris-setosa
9	8	5	3.4	1.5	0.2	Iris-setosa
10	9	4.4	2.9	1.4	0.2	Iris-setosa
11	10	4.9	3.1	1.5	0.1	Iris-setosa
12	11	5.4	3.7	1.5	0.2	Iris-setosa
13	12	4.8	3.4	1.6	0.2	Iris-setosa
14	13	4.8	3	1.4	0.1	Iris-setosa



Data

bank_df_s2 45209 obs. of 17 variables

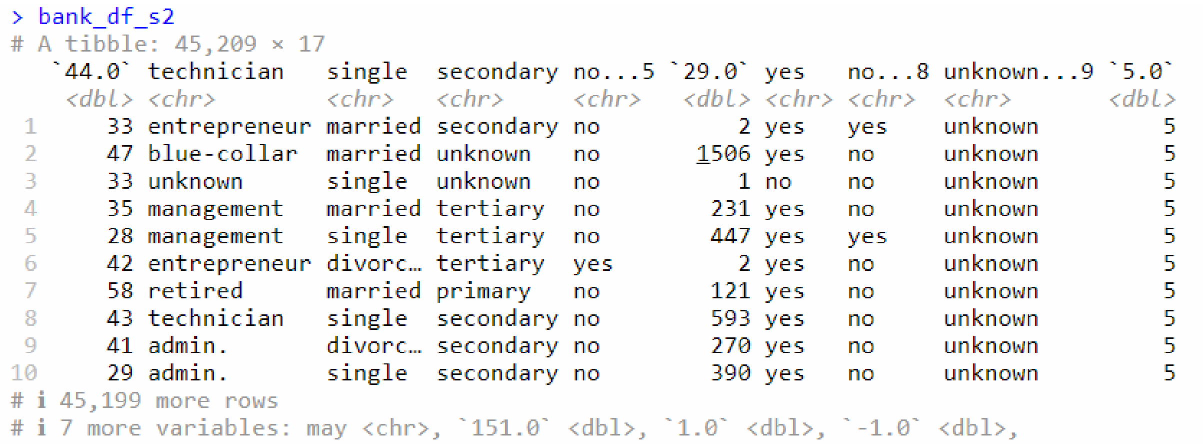
iris 150 obs. of 6 variables

Reading Specific Rows

Let's read specific rows from a workbook by setting the skip and n_max arguments. For skipping the first few rows, you can use the skip argument with a value equal to the number of rows you want to skip.

bank_df_s2 <- read_excel("sample-dataset 2-3.xlsx", sheet = "bank-full", skip = 2)

Please note that the above code skips the headers as well.



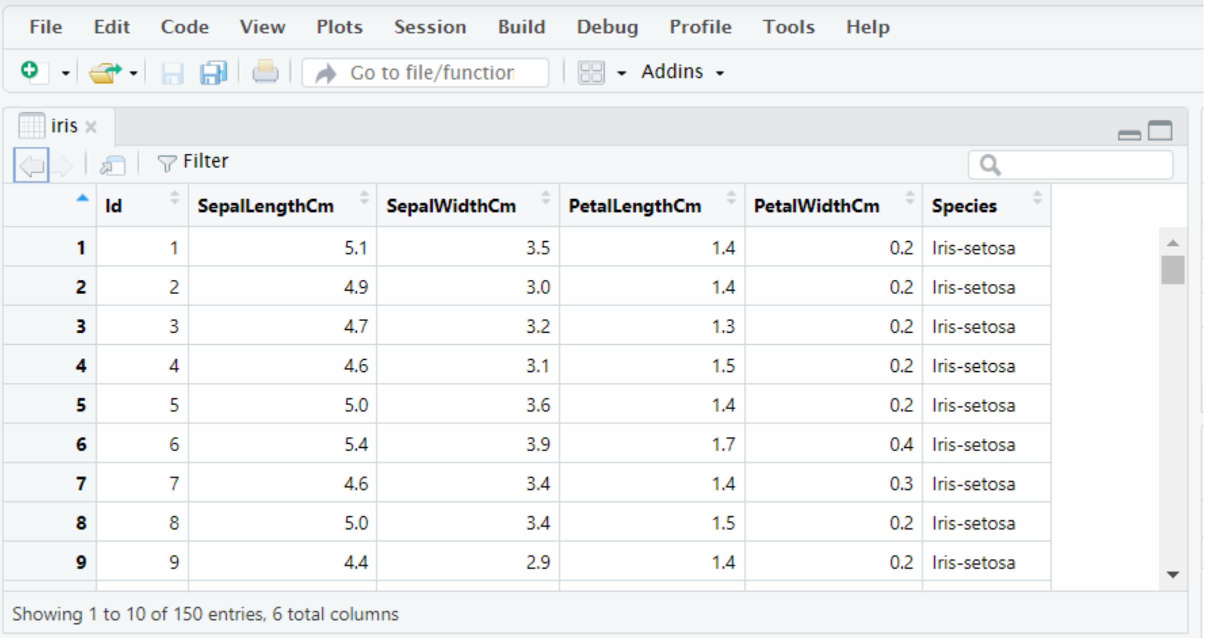
```
> bank_df_s2
# A tibble: 45,209 × 17
   `44.0` technician   single  secondary no...5 `29.0` yes   no...8 unknown...9 `5.0`
    <dbl> <chr>        <chr>   <chr>     <chr>   <dbl> <chr> <chr>  <chr>       <dbl>
 1     33 entrepreneur married secondary no          2 yes   yes    unknown         5
 2     47 blue-collar  married unknown   no       1506 yes   no     unknown         5
 3     33 unknown      single  unknown   no          1 no    no     unknown         5
 4     35 management   married tertiary  no        231 yes   no     unknown         5
 5     28 management   single  tertiary  no        447 yes   yes    unknown         5
 6     42 entrepreneur divorc… tertiary  yes         2 yes   no     unknown         5
 7     58 retired      married primary   no        121 yes   no     unknown         5
 8     43 technician   single  secondary no        593 yes   no     unknown         5
 9     41 admin.       divorc… secondary no        270 yes   no     unknown         5
10     29 admin.       single  secondary no        390 yes   no     unknown         5
# ℹ 45,199 more rows
# ℹ 7 more variables: may <chr>, `151.0` <dbl>, `1.0` <dbl>, `-1.0` <dbl>,
```

Step 4: Viewing the Data

After reading an Excel file, you probably want to view the data. You can do this using the print() function.

# Print the data
print(data)
Alternatively, you can use the View() function to open your data in a spreadsheet-like format.

# View the data
View(data) *//V is Capital*



	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
1	1	5.1	3.5	1.4	0.2	Iris-setosa
2	2	4.9	3.0	1.4	0.2	Iris-setosa
3	3	4.7	3.2	1.3	0.2	Iris-setosa
4	4	4.6	3.1	1.5	0.2	Iris-setosa
5	5	5.0	3.6	1.4	0.2	Iris-setosa
6	6	5.4	3.9	1.7	0.4	Iris-setosa
7	7	4.6	3.4	1.4	0.3	Iris-setosa
8	8	5.0	3.4	1.5	0.2	Iris-setosa
9	9	4.4	2.9	1.4	0.2	Iris-setosa

Showing 1 to 10 of 150 entries, 6 total columns

Used-View(iris)command

Step 5: Handling Multiple Sheets

If your Excel file contains multiple sheets, you can specify the sheet you want to read using the 'sheet' argument in the read_excel() function.

```
# Read a specific sheet

data <- read_excel("path/to/your/file.xlsx", sheet = "Sheet2")
```

Inclass Assignment:

1. Reading Specific Cells
2. Skipping Columns

Reference Link:

https://egyankosh.ac.in/bitstream/123456789/87562/1/Unit-14.pdf