Data Import / Export

Day 2

Agenda – Day 2

- Data Import/Export Overview
- Need for data import/export in R
- Checks before data import
- Commonly encountered format
- Import functions in R
  - ✓ scan() – Import function
  - ✓ read.table () – Import functions
  - ✓ read.delim () – Import function
  - ✓ read.csv () – Import function
- Dealing with missing values
- Export functions in R
  - ✓ write.table () – Export functions

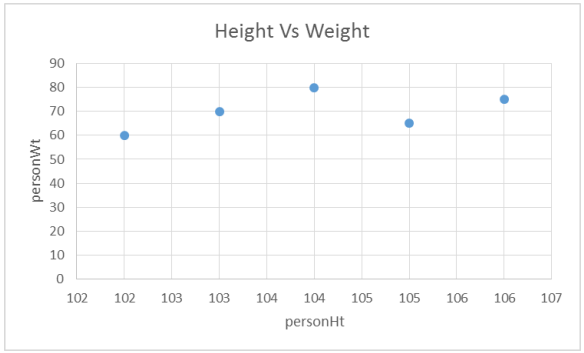
Need for data import in R

Let us consider that, we have two numeric vectors personHt, personWt and each numeric vector have 5 values each as shown below:

```
> personHt = c(102,103,104,105,106)
> personWt = c(60,70,80,65,75)
```

If we want to do any bivariate analysis, we can create a dataframe with the name person by combining personHt and personWt

```
> person=data.frame(personHt,personWt)
> person
  personHt personWt
1      102       60
2      103       70
3      104       80
4      105       65
5      106       75
```



- ✓ This kind of process works well when we have tiny bits of data but, if we have millions of rows this approach is not a practical approach
- ✓ We can expect certain issues to creep in all the time due to typos
- ✓ In general it is highly recommended to use data import / export options

Checks before data import

When reading data from external files (text file, comma separated files,…etc.) it is necessary to know and understand more about the inherent characteristics

- ✓ Presence of header line
- ✓ Kind of value separator
- ✓ Representation of missing values
- ✓ Existence of any blank lines
- ✓ Classes of the variables (i.e. variable type)

Commonly encountered format

Plain text files are the commonly encountered formats in data science project

It generally contains table like format with headers in the first row and the corresponding values in the remaining rows. All the values in the text files will be separated by delimiters like comma , tab , semicolon, pipe (or a vertical bar), space

Table – structured csv data with headers

```
Name,age,gender,height,weight
John,45,M,175,75
James,55,M,195,87
Peter,23,M,150,82
Sarah,45,F,167,62
```

Header row consist of column names in this case separated by Comma

The data rows are in the same format as the header rows, but each rows contain actual data values

By using Import functions we can load the above text files into R as data.frame structure which exactly contains the below structure.

	A	B	C	D	E
1	Name	age	gender	height	weight
2	John	45	M	175	75
3	James	55	M	195	87
4	Peter	23	M	150	82
5	Sarah	45	F	167	62

Please Note : While working is R it is easy to apply statistical models in data.frame structures

Import functions in R

Different methods of reading data in R

- ✓ scan : Reads vectors of data which all have the same mode (i.e variable type)
- ✓read.table : Create data frames with different mode columns (variables)
- ✓read.delim : reads delimited files into data frame object
- ✓read.csv : reads csv files into data frame object

We will learn more about these import functions in consecutive slides

scan() – Import function

Scan() reads the field of data in the file as specified by the what option, with the default being numeric

The entire syntax of scan () look like this:

Name of the file to read value from

Type of data to read

Maximum no of values to read

Character which delimit field

Number of lines to skip before reading

```
scan(file = "", what = double(), nmax = -1, n = -1, sep = "",
     quote = if(identical(sep, "\n")) "" else "'\"", dec = ".",
     skip = 0, nlines = 0, na.strings = "NA",
     flush = FALSE, fill = FALSE, strip.white = FALSE,
     quiet = FALSE, blank.lines.skip = TRUE, multi.line = TRUE,
     comment.char = "", allowEscapes = FALSE,
     fileEncoding = "", encoding = "unknown", text, skipNul = FALSE)
```

As you can see there are lot of parameters which goes into the scan function but, for more general usage we would require only few of them (as highlighted above)

Advantages of scan() function

✓Reads data directly from the console

✓Reads data from external file

read.table () – Import function

read.table () reads a file in table format and creates a data frame from it

The entire syntax of read.table() look like this:

Name of the file to read value from

Logical (T/F), refers to first line of file as header row

Character which delimits field

A vector of optional names for the variables

Maximum no of values to read

Maximum no of values to read

```
read.table(file, header = FALSE, sep = "", quote = "\"'",
           dec = ".", numerals = c("allow.loss", "warn.loss", "no.loss"),
           row.names, col.names, as.is = !stringsAsFactors,
           na.strings = "NA", colClasses = NA, nrows = -1,
           skip = 0, check.names = TRUE, fill = !blank.lines.skip,
           strip.white = FALSE, blank.lines.skip = TRUE,
           comment.char = "#",
           allowEscapes = FALSE, flush = FALSE,
           stringsAsFactors = default.stringsAsFactors(),
           fileEncoding = "", encoding = "unknown", text, skipNul = FALSE)
```

As you can see there are lot of parameters which goes into the scan function but, for more general usage we would require only few of them

Advantages of read.table() function

✓read.table () reads a file in table format and creates a data frame from it which helps to do statistical analysis

read.delim () – Import function

Simply a wrapper function for read.table() with default argument values useful for reading in tab-separated data

The entire syntax of read.delim() look like this:

Name of the file to read value from

Logical (T/F), refers to first line of file as header row

Default is tab delimiter

```
read.delim(file, header = TRUE, sep = "\t", quote = "\"",
           dec = ".", fill = TRUE, comment.char = "", ...)
```

It also provides additional function called read.delim2() which would vary on one particular parameter.

```
read.delim2(file, header = TRUE, sep = "\t", quote = "\"",
            dec = ",", fill = TRUE, comment.char = "", ...)
```

The character used in the file for decimal points

We can see that in read.delim2() the decimal parameter is said to comma (which is most common practice in some countries)

Advantages of read.delim() function

✓ read.table () provides support for other kind of delimiters like comma, space, pipe present in the text file

read.csv () – Import function

Simply a wrapper function for read.table() with default argument values useful for reading in comma-separated data

The entire syntax of read.csv() look like this:

Name of the file to read value from

Logical (T/F), refers to first line of file as header row

Default is comma delimiter

```
read.csv(file, header = TRUE, sep = ",", quote = "\"",
         dec = ".", fill = TRUE, comment.char = "", ...)
```

It also provides additional function called read.csv2() which would vary on two parameters

```
read.csv2(file, header = TRUE, sep = ";", quote = "\"",
          dec = ",", fill = TRUE, comment.char = "", ...)
```

The character used in the file for decimal points

We can also see that in read.csv2() the decimal parameter is said to comma (which is most common practice in some countries) and delimiter is said to be ";"

Advantages of read.csv() function

✓ read.csv () provides support for other kind of delimiters like comma, space, pipe present in the text file

Dealing with missing values

R import function read any missing values in input source file with a NA representation certain aspect in this vary depends on the type of variables (Numeric / Character)

✓Generally empty fields in numeric variables are treated as missing

✓For character variables, empty fields are not considered as missing by default

- Input source file should have NA coded in place of empty fields
- This point is a kind of limitations for reading plain text files
- This option can be customized by using na.strings() parameter in read.table function

Export functions in R

Different methods of writing data in R

write.table : writes data frames into text files of different delimiter formats

write.csv : writes data frame objects into csv file

The entire syntax of write.table() look like this:

The R object to be written

Character string naming a file path

Character which delimits fields

```
write.table(x, file = "", append = FALSE, quote = TRUE, sep = " ",
            eol = "\n", na = "NA", dec = ".", row.names = TRUE,
            col.names = TRUE, qmethod = c("escape", "double"),
            fileEncoding = "")

write.csv(...)
write.csv2(...)
```

Logical (T/F), whether row names to be returned in file

write.csv and write.csv2 provide convenience wrapper for writing csv files, variations present for sep and dec parameters similar to read.csv and read.csv2 functions

Appendix

Flow of presentation

Delimiter Formats

A **delimiter** is a sequence of one or more characters used to specify the boundary between separate, independent regions in plain text or other data streams

Note : Any character can be used as a delimiter but, most commonly used delimiters are comma,tab,colon,pipe (or vertical bar), space