Lecture 10. Reading and Writing Files

R and Data Visualization

BIG2006, Hanyang University, Fall 2022

File Handling

Loading and saving data in an active workspace by reading and writing files.

Typically, to work with a large data set, you will need to read in the data from an external file, whether it is stored as plain text, in a spread sheet file, or on a website.

Creating a File

- `file.create()`: create a new file from console or truncate if already exists
- The function returns a TRUE logical value if file is created, otherwise returns FALSE.

```
file.create("Lec10_tmp/BIG.txt")
```

```
## [1] TRUE
```

Writing into a File

- `write.table()`: write an object to a file
- The function is present in `utils` package and write data frame or matrix object to any type of file.

```
# Write iris dataset into the text file
write.table(x = iris[1:15,],
            file = "Lec10_tmp/BIG.txt")
# read.table("BIG.txt")[1:5,1:3]
```

Reading a File

- `read.table()`: read a file and show output as data frame
- The function helps in analyzing the data frame for further computations.

```
new.iris <- read.table(file = "Lec10_tmp/BIG.txt")
print(new.iris[1:5,1:3])
```

```
##   Sepal.Length Sepal.Width Petal.Length
## 1          5.1         3.5          1.4
## 2          4.9         3.0          1.4
## 3          4.7         3.2          1.3
## 4          4.6         3.1          1.5
## 5          5.0         3.6          1.4
```

Renaming a File

- `file.rename()`: renames the file and return a logical value
- The function renames files but not directories.

```
file.rename(from = "Lec10_tmp/BIG.txt",
            to = "Lec10_tmp/newBIG.txt")
```

```
## [1] TRUE
```

Check Existence of a File

- `file.exists()`: check whether a file exists or not in current working directory
- The function returns TRUE local value if file exists, otherwise returns FALSE.

```
file.exists("Lec10_tmp/BIG.txt")
```

```
## [1] FALSE
```

```
file.exists("Lec10_tmp/newBIG.txt")
```

```
## [1] TRUE
```

List All Files

- `list.files()`: shows all files of specified path
- If path is not passed in the function parameter, files present in current working directory is shown as output.

```
list.files("Lec10_tmp")
```

```
## [1] "bigdata.txt"         "mydatafile.txt"      "newBIG.txt"
## [4] "spreadsheetfile.csv"
```

Copy a File and Create a Directory

- `file.copy()`: create a copy of specified file from console itself
- `dir.create()`: create a directory in the path specified in the function parameter

```
file.copy(from = "Lec10_tmp/newBIG.txt",
          to = "Lec10_tmp/newBIG2.txt")
```

```
## [1] TRUE
```

```
#dir.create("BIG")
```

Reading in External Data Files

The Table Format

- Table-format files are best thought of as plain-text files with three key features:
  1. **Header**: It provides names for each column of data. If a header is present, it is always the first line of the file.
  2. **Delimiter**: A character used to separate the entries in each line
  3. **Missing value**: When reading the file, R will turn these entries into the form it recognizes, NA.
- Typically, these files have a `.txt` extension or `.csv` (for comma-separated values).

Note: The first line is the header, the values are delimited with a single space, and missing values are denoted with an asterisk (*). Also, each new record is required to start on a new line.

```
mydatafile <- read.table(file = "Lec10_tmp/mydatafile.txt",
                         header = TRUE, sep = " ",
                         na.strings = "*",
                         stringsAsFactors = FALSE)
mydatafile
```

```
##   person age gender funny age.mon
## 1  Peter  NA      M  High     504
## 2   Lois  40      F  <NA>     480
## 3    Meg  17      F   Low     204
## 4  Chris  14      M   Med     168
## 5 Stewie   1      M  High      NA
## 6  Brian  NA      M   Med      NA
```

Note: The `file.choose()` command opens your filesystem viewer directly from the R prompt. Then, you can navigate to the folder of interest, and after you select your file, only a character string is returned.

- `stringsAsFactors=FALSE` prevents R from treating all non-numeric elements as factors.
- You can overwrite `gender` and `funny` with factor versions of themselves.

```
mydatafile$gender <- as.factor(mydatafile$gender)
mydatafile$funny <- factor(x = mydatafile$funny,
                           levels = c("Low","Med","High"))
mydatafile
```

```
##   person age gender funny age.mon
## 1  Peter  NA      M  High     504
## 2   Lois  40      F  <NA>     480
## 3    Meg  17      F   Low     204
## 4  Chris  14      M   Med     168
## 5 Stewie   1      M  High      NA
## 6  Brian  NA      M   Med      NA
```

Spreadsheet Workbooks

- You should first convert the spreadsheet to a table format. In Excel, save the spreadsheet as a comma-separated file.
- R has a shortcut version of `read.table`, `read.csv`, for these files.

```
spread <- read.csv(file = "Lec10_tmp/spreadsheetfile.csv",
                   header = FALSE, stringsAsFactors = TRUE)
spread[1:5,]
```

```
##   V1  V2     V3
## 1 55 161 female
## 2 85 185   male
## 3 75 174   male
## 4 42 154 female
## 5 93 188   male
```

Web-Based Files

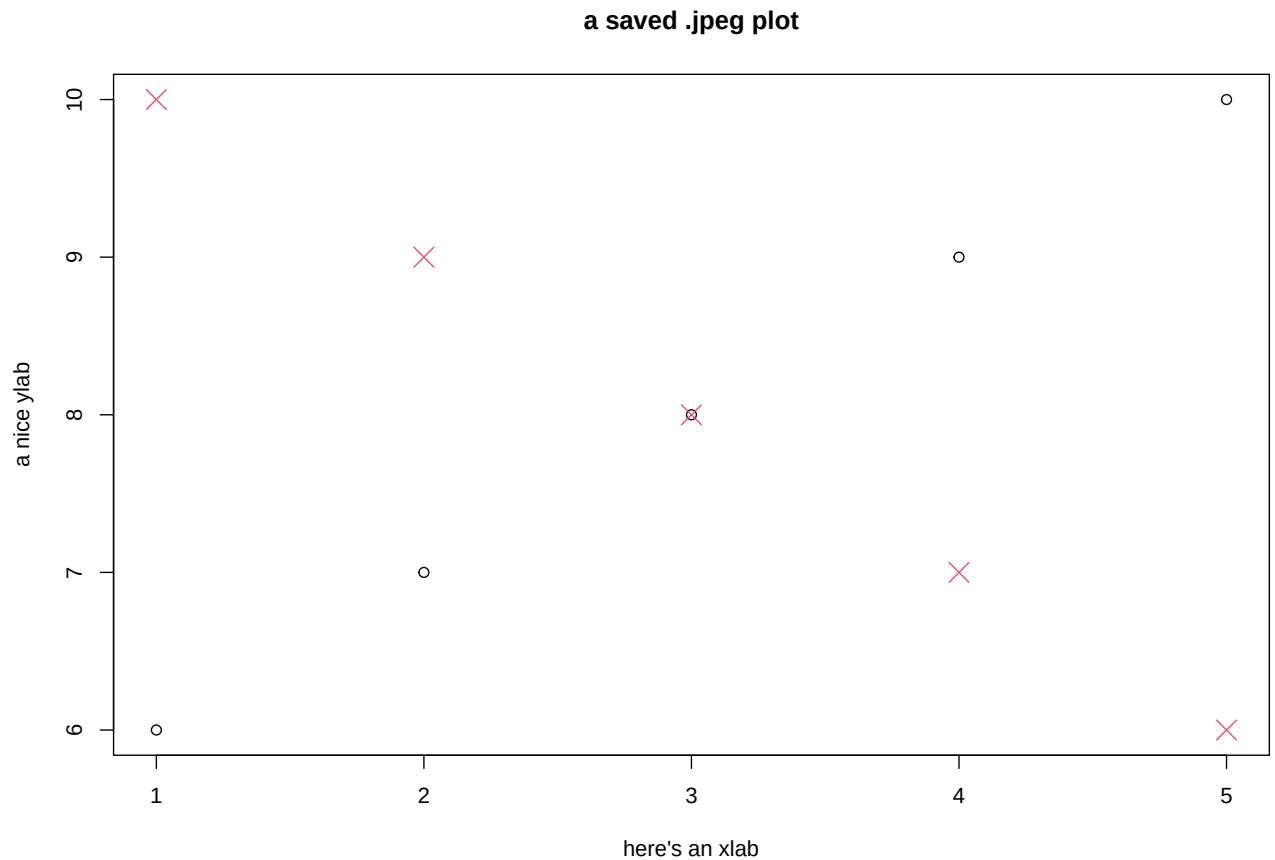
- R can read in files from a website with the same `read.table` command.
- You just have to specify the URL address of the file instead of a local folder location.

```
url <- "http://courses.washington.edu/b517/Datasets/string.txt"
data <- read.table(url, header = TRUE)
head(data)
```

```
##    x       y
## 1 10 34.7081
## 2 12 34.5034
## 3 14 36.5656
## 4 16 38.3125
## 5 18 42.5441
## 6 20 43.7210
```

Plots and Graphics Files

Plots can be written directly to a file. R support direct writing to `.jpeg`, `.bmp`, `.png`, and `.tiff` files.



```
jpeg(filename = "Lec10_tmp/myjpegplot.jpeg",
     width = 600, height = 600)
plot(1:5, 6:10, ylab = "a nice ylab",
     xlab = "here's an xlab",
     main = "a saved .jpeg plot")
points(1:5, 10:6, cex = 2, pch = 4, col = 2)
dev.off()
```

```
## pdf
##   2
```

```
pdf(file = "Lec10_tmp/myjpegplot.pdf",
     width = 5, height = 5)
plot(1:5, 6:10, ylab = "a nice ylab",
     xlab = "here's an xlab",
     main = "a saved .jpeg plot")
points(1:5, 10:6, cex = 2, pch = 4, col = 2)
dev.off()
```

```
## pdf
##   2
```

Ad Hoc Object Read/Write Operations

If you need to read or write R objects, such as lists or arrays, you will need the `dput` and `dget` commands, which can handle objects in a more ad hoc style.

These commands have some drawbacks, e.g., `dput` is not as reliable a command as `write.table` to create the necessary plain-text representation for an object and it needs to store structural information.

Nevertheless, they are useful ways to store or transfer specific objects without having to save an entire workspace.

```r
somelist <- list(foo=c(5,2,45),
            bar=matrix(data=c(T,T,F,F,F,F,T,F,T),nrow=3,ncol=3),
            baz=factor(c(1,2,2,3,1,1,3),levels=1:3,ordered=T))
somelist
```

```
## $foo
## [1]  5  2 45
##
## $bar
##       [,1]  [,2]  [,3]
## [1,]  TRUE FALSE  TRUE
## [2,]  TRUE FALSE FALSE
## [3,] FALSE FALSE  TRUE
##
## $baz
## [1] 1 2 2 3 1 1 3
## Levels: 1 < 2 < 3
```

- dput: store the object as a plain-text file
- dget: read the object from the myRobject.txt file

```
dput(x = somelist, file="Lec10_tmp/myRobject.txt")
newobject <- dget(file="Lec10_tmp/myRobject.txt")
newobject
```

```
## $foo
## [1]  5  2 45
##
## $bar
##       [,1]  [,2]  [,3]
## [1,]  TRUE FALSE  TRUE
## [2,]  TRUE FALSE FALSE
## [3,] FALSE FALSE  TRUE
##
## $baz
## [1] 1 2 2 3 1 1 3
## Levels: 1 < 2 < 3
```

Some Important Function/Operator

- `read.table`: Import table-format data file
- `list.files`: Print specific folder contents
- `file.choose`: Interactive file selection
- `read.csv`: Import comma-delimited file
- `write.table`: Write table-format file to disk
- `jpeg`, `bmp`, `png`, `tiff`: Write image/plot file to disk
- `dev.off`: Close file graphics device
- `pdf`, `postscript`: Write image/plot file to disk
- `dput`: Write R object to file (ASCII)
- `dget`: Import ASCII object file

Reference

- Davies, T. M. The Book of R. No Starch Press. Chapter 8.