Lab 05 - Reading Files and Web Scrapping using R

Best practice — Reproducible data science

- Manual Data Change Log
  - Record where to obtain the original data
  - Use a file such as *CHANGELOG.txt* to make dated notes in reverse chronological order
  - Make a copy of the entire project whenever significant changes are made
- Data Version control
  - Use a *Data Science Version Control System* (https://dvc.org/)
  - Use a *Version Control System* (git, subversion and etc.) for managing different versions of code.

See https://www.datacamp.com/community/blog/version-control-data-science

Reading data from flat files

Flat files are plain text files with rows of columns separated by a delimiter.

For example, comma or semi-colon separated values (`.csv`), tab-separated values (`.tsv`), pipe-separated files, are all considered as flat files.

R's built-in base function `read.table()` can be used to read most separated value formats. Depending on the data file format, additional arguments may need to be appropriately set so that the file is read in correctly.

For example, using `read.table()` to read in a data file whose name is stored in the variable `file`:

```
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)
```

`read.table` is not the right tool for reading large matrices, especially those with many columns: it is designed to read data frames which may have columns of very different classes.

To read large matrices, use `scan()` instead.

Variations of `read.table()`

Instead of `read_table()`, the following R functions can be used also:

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

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

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

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

Note that these functions can consume a surprising amount of memory when reading large files.

Example 1.

Comma Separated Values (csv), using the base `read.csv()`:

The IRIS dataset from UCI: http://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data is a *comma separated values* (*csv*) file containing the length and width values of the petal and sepal of 3 iris species, each of which has 50 observations.

Using the `read.csv()` function:

```
path <- url("http://archive.ics.uci.edu/ml/machine-learning-databases/iris/iri
col.names <- c("sepal_length", "sepal_width",
               "petal_length", "petal_width",
               "class")
iris_data <- read.csv(path, col.names = col.names)
head(iris_data)
##   sepal_length sepal_width petal_length petal_width       class
## 1          4.9         3.0          1.4         0.2 Iris-setosa
## 2          4.7         3.2          1.3         0.2 Iris-setosa
## 3          4.6         3.1          1.5         0.2 Iris-setosa
## 4          5.0         3.6          1.4         0.2 Iris-setosa
## 5          5.4         3.9          1.7         0.4 Iris-setosa
## 6          4.6         3.4          1.4         0.3 Iris-setosa
```

There is a problem with the above code, can you spot it?

Example 2.

Try the code below and inspect if the data has been read in correctly.

```
path <- url("https://assets.datacamp.com/production/course_1477/datasets/hotdo
hotdogs <- read.table(path,
                      sep = "",
                      col.names = c("type", "calories", "sodium"))
head(hotdogs)
##   type calories sodium
## 1 Beef      186    495
## 2 Beef      181    477
## 3 Beef      176    425
## 4 Beef      149    322
## 5 Beef      184    482
## 6 Beef      190    587
```

Reading from Excel

Download the files *urbanpop.xlsx* and *urbanpop.xls* and save them somewhere on your hard drive. Set the `path` variable below appropriately, e.g., `path <- "C:/CITS4009/labs/data/"`. The relative path (relative to the work directory that you set) can also be used, like below:

```
path <- "../data/"
```

```
# This package contain the functions for reading both .xls and .xlsx files.
library(readxl)

workbook <- paste0(path, "urbanpop.xlsx")
# By default, the first workbook would be read.
df <- read_excel(workbook, sheet="1975-2011")
head(df)
```

```
## # A tibble: 6 × 38
##   country  `1975` `1976` `1977` `1978` `1979` `1980` `1981` `1982` `1983` `
##   <chr>     <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>
## 1 Afghani… 1.79e6 1.91e6 2.02e6 2.14e6 2.27e6 2.40e6 2.49e6 2.59e6 2.69e6 2
## 2 Albania  7.85e5 8.08e5 8.31e5 8.54e5 8.78e5 9.02e5 9.27e5 9.52e5 9.78e5 1
## 3 Algeria  6.46e6 6.77e6 7.10e6 7.45e6 7.81e6 8.19e6 8.64e6 9.11e6 9.59e6 1
## 4 America… 2.16e4 2.20e4 2.25e4 2.29e4 2.35e4 2.42e4 2.52e4 2.63e4 2.77e4 2
## 5 Andorra  2.70e4 2.84e4 2.97e4 3.10e4 3.26e4 3.44e4 3.64e4 3.86e4 4.10e4 4
## 6 Angola   1.27e6 1.37e6 1.48e6 1.60e6 1.72e6 1.86e6 2.02e6 2.19e6 2.37e6 2
## # … with 27 more variables: `1985` <dbl>, `1986` <dbl>, `1987` <dbl>,
## #   `1988` <dbl>, `1989` <dbl>, `1990` <dbl>, `1991` <dbl>, `1992` <dbl>,
## #   `1993` <dbl>, `1994` <dbl>, `1995` <dbl>, `1996` <dbl>, `1997` <dbl>,
## #   `1998` <dbl>, `1999` <dbl>, `2000` <dbl>, `2001` <dbl>, `2002` <dbl>,
## #   `2003` <dbl>, `2004` <dbl>, `2005` <dbl>, `2006` <dbl>, `2007` <dbl>,
## #   `2008` <dbl>, `2009` <dbl>, `2010` <dbl>, `2011` <dbl>
## # ℹ Use `colnames()` to see all variable names
```

By default, the first worksheet will be read. If you want to read a different worksheet, then use the `sheet` argument, e.g., `sheet=2` is for reading the second worksheet; or `sheet="1967-1974"` if `1967-1974` is the name of the desired worksheet.

Try modifying the code above to read in the `urbanpop.xls` file. Inspect the data files (using Excel outside R) and the data frames. Have you read in the data correctly?

Web scrapping using R

You can call appropriate R functions to extract information from a web page. For the optional exercise here, firstly open a web browser and go to the URL given below to view the content of the web page, then try the following R statements (you may need to substitute `<i>` by `<I>` and `</i>` by `</I>`):

```
library(dplyr)

# Get the page's source
web_page <- readLines("http://www.programmingr.com/jan09rlist.html")
```

```
## Warning in readLines("http://www.programmingr.com/jan09rlist.html"): incomp
## final line found on 'http://www.programmingr.com/jan09rlist.html'
```

```
# Note that the web page above is quite complicated. You need to know its
# contents so that you can extract the correct pattern for the lines that
# are of interest to you.

# We notice that the authors' names are all in italics. So we pull out
# those lines containing "<i>" which is the HTML tag for the beginning
# of italics font.
author_lines <- web_page[grep("<I>", web_page)]
# We need to discard some lines before and after those lines we got above.
author_lines <- author_lines[1:2994]

# Delete the unwanted characters in the lines we pulled out
authors <- gsub("<I>", "", author_lines) %>%
    gsub("</i>.*", "", .)

# Present only the ten most frequent posters
author_counts <- sort(table(authors), decreasing = TRUE)
author_counts[1:10]
```

```
## authors
##   Gabor Grothendieck      David Winsemius       Duncan Murdoch
##                  116                   93                   84
##    Prof Brian Ripley          jim holtman    Wacek Kusnierczyk
##                   84                   80                   55
##        Marc Schwartz       hadley wickham Henrique Dallazuanna
##                   48                   40                   36
##            Greg Snow
##                   35
```

Previous
Lab 04 - EDA for Multiple Variables

Next
Lab 06 - Explain ML Models using LIME

Last updated 2 months ago