Lab2 Instruction

Jun Cai

September 18, 2015

Note

- All R codes are present in boxes with grey background. You can run them in your R Console. Lines leading by `##` are outputs of R codes.
- All R codes are run correctly on my own Mac OS X. When you try them on your own computer, please customize your own working directory.
- The functions in **bold** are recommended to use in your data analysis.
- The Lab Instruction in different formats including .Rmd, .md, .html and .pdf are available on my GitHub. For your convenience, the data used and results produced in the Lab are also provided in the Github.

Lab2 contains basics for file and directory manipulation, and R data input and output. Rather than a complete collection of functions, I will introduce the frequenty-used functions from my own R experience.

Part I File and Directory Manipulation

R has a variety of functions for file and directory manipulation. The following are a few examples:

`setwd()` and `getwd()`: used to change or determine the current working directory. It's a good habit to set working directory before your data analysis as all results during your data analysis will be stored in the working directory.

`list.files()` and `list.dirs()`: returns a character vector of names of files or directories under the given directory.

`file.info()`: gives file size, creation time, directory vs. ordinary file status, and so on for each file whose name is in the argument, a character vector.

`file.create()` and `dir.create()`: creates files or directories with the given names if they do not already exist.

`file.exists()` and `dir.exists()`: returns a logical vector indicating whether the given file exists for each name in the first argument, a character vector.

`file.copy()` and `file.rename()`: moves files `from` source path `to` destination path.

`file.remove()` and **`unlink()`**: deletes the files or directories specified by the first argument, a character vector.

```r
# set current working directory to DAE
setwd("/Users/tonytsai/Documents/R/DAE")
getwd()
```

```
## [1] "/Users/tonytsai/Documents/R/DAE"
```

```r
# list all files including directories under current working directory DAE
list.files()
```

```
## [1] "Lab1note.pdf"             "Lab2"
## [3] "Lecture1Introduction.pdf" "LICENSE"
## [5] "README.md"                "reference"
## [7] "script"                   "生态数据分析课程大纲.pdf"
```

```r
# extract file information for those files
file.info(list.files())
```

```
##                             size isdir mode               mtime
## Lab1note.pdf              553464 FALSE  777 2015-09-20 12:59:30
## Lab2                         306  TRUE  755 2015-09-21 10:13:57
## Lecture1Introduction.pdf 1134795 FALSE  640 2015-09-20 12:59:16
## LICENSE                     1077 FALSE  644 2015-09-18 09:40:04
## README.md                    244 FALSE  644 2015-09-18 09:58:11
## reference                    170  TRUE  777 2015-09-16 11:22:01
## script                       136  TRUE  755 2015-09-21 10:13:47
## 生态数据分析课程大纲.pdf  253722 FALSE  777 2015-09-12 18:07:12
##                                        ctime               atime uid gid
## Lab1note.pdf             2015-09-20 14:24:21 2015-09-20 14:24:19 501  20
## Lab2                     2015-09-21 10:13:57 2015-09-21 10:13:47 501  20
## Lecture1Introduction.pdf 2015-09-20 12:59:29 2015-09-20 12:59:15 501  20
## LICENSE                  2015-09-18 09:42:06 2015-09-20 12:49:52 501  20
## README.md                2015-09-18 09:58:11 2015-09-18 09:58:04 501  20
## reference                2015-09-16 13:27:58 2015-09-21 10:13:47 501  20
## script                   2015-09-21 10:13:47 2015-09-21 10:13:47 501  20
## 生态数据分析课程大纲.pdf 2015-09-16 13:27:58 2015-09-16 13:27:58 501  20
##                             uname grname
## Lab1note.pdf             tonytsai  staff
## Lab2                     tonytsai  staff
## Lecture1Introduction.pdf tonytsai  staff
## LICENSE                  tonytsai  staff
## README.md                tonytsai  staff
## reference                tonytsai  staff
## script                   tonytsai  staff
## 生态数据分析课程大纲.pdf tonytsai  staff
```

```r
# list only directories under DAE
list.dirs()
```

```
## [1] "."                  "./Lab2"             "./Lab2/data"
## [4] "./Lab2/data/CMDSSS" "./reference"        "./script"
```

```r
# find all R scripts under DAE and give their full path names (or absolute paths)
list.files(recursive = TRUE, pattern = ".R$", full.names = TRUE)
```

```
## [1] "./script/20150916.R"
```

```
# create a recursive directory under DAE/Lab2, which stores the TXT data
# that will be read in Part III.
if(!dir.exists("Lab2/data/CMDSSS"))
  dir.create("Lab2/data/CMDSSS", recursive = TRUE)

# create a temporary directory under script
if(!dir.exists("script/tmp")) dir.create("script/tmp")
# create a temporary R script under tmp to say Hello World, Hello R!
file.create("script/tmp/tmp.R")
```

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

```
cat("print('Hello World, Hello R!')", file = "script/tmp/tmp.R")
# excute the R script
source("script/tmp/tmp.R")
```

```
## [1] "Hello World, Hello R!"
```

```
# copy tmp.R to helloworld.R
file.copy("script/tmp/tmp.R", "script/helloworld.R")
```

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

```
list.files("script", recursive = TRUE)
```

```
## [1] "20150916.R"   "helloworld.R" "tmp/tmp.R"
```

```
# rename helloworld.R to hello.R
file.rename("script/helloworld.R", "script/hello.R")
```

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

```
list.files("script", recursive = TRUE)
```

```
## [1] "20150916.R" "hello.R"    "tmp/tmp.R"
```

```
# delete all R scripts under script directory except for 20150916.R
# attempt to delete inexistent hellworld.R
file.remove(c("script/hello.R", "script/helloworld.R"))
```

```
## Warning in file.remove(c("script/hello.R", "script/helloworld.R")): cannot
## remove file 'script/helloworld.R', reason 'No such file or directory'
```

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

```
list.files("script", recursive = TRUE)
```

```
## [1] "20150916.R" "tmp/tmp.R"
```

```
# delete the temporary directory that is not empty.
unlink("script/tmp", recursive = TRUE)
list.files("script")
```

```
## [1] "20150916.R"
```

To see all the file- and directory-related functions, type the following:

```
?files
```

Part II Capturing R Console Output

R provides functions to save the results that appear in your R Console into a text file.

`sink()`: diverts R output to a file connection.

`capture.output()`: sends R output to a character string or file connection.

```
# type following codes in your Console
# divert R output to CO2.txt under data directory
sink(file = "data/CO2.txt")
# load R built-in dataset CO2. type ?CO2 to see the decription about CO2 dataset.
data("CO2")
# divert the first 14 rows of CO2
head(CO2, 14)
# end the divertion
sink()
```

```
# sink() does not work in knitr, because it is already used internally to capture
# results. To make it work, you have to use {} wrap up all aboving code in a
# single expression and print results to Console explicitly.
{
  sink(file = "data/CO2.txt")
  data("CO2")
  print(head(CO2, 14))
  sink()
}
# the content of CO2.txt will be checked in Part III.
```

```
# generate a vector containing a numeric sequence from 1 to 10
1:10
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10
```

```
# send the output to variable x
x <- capture.output(1:10)
x
```

```
## [1] " [1]  1  2  3  4  5  6  7  8  9 10"
```

```r
# divert the first 14 rows of CO2 dataset to CO2.txt
capture.output(head(CO2, 14), file = "data/CO2.txt")
```

Part III Imports and Exports

Here I list some functions in R for reading and storing data in common formats, such as Plain Text (.txt), CSV (.csv), Excel (.xls, .xlsx), and dBase (.dbf). For large and complex data, databases are supposed to be used for management.

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

`read.csv()` and **`write.csv()`**: reads a CSV file and creates a data frame from it. A standard CSV file actually stores tabular data in plain text with values separated by comma.

`read.xlsx()` and `write.xlsx()`: reads and writes Excel 2007 and Excel 97/2000/XP/2003 files. They are from xlsx package. There are some other packages for reading or/and Excel files, such as openxlsx, XLConnect, readxl and WriteXLS. Frankly, I am not used to reading and writing Excel files. I usually process Excel files by saving them as CSV files.

`read.dbf()` and `write.dbf()`: reads and writes dBase files. They are from foreign package, which provides functions for reading and writing data stored by Minitab, S, SAS, SPSS, Stata, dBase, ... The reason why I specially introduce how to read dBase files is that the attribute table of shape file is stored in .dbf format.

`save()` and **`load()`**: writes R objects to the specified file. The objects can be read back from the file later by using `load`.

`data()`: loads specified data sets, or list the available data sets.

```r
# read data in Plain Text
txt <- read.table(file = "data/CO2.txt")
# check the content of CO2.txt
str(txt)
```

```
## 'data.frame':    14 obs. of  5 variables:
##  $ Plant    : Factor w/ 2 levels "Qn1","Qn2": 1 1 1 1 1 1 1 2 2 2 ...
##  $ Type     : Factor w/ 1 level "Quebec": 1 1 1 1 1 1 1 1 1 1 ...
##  $ Treatment: Factor w/ 1 level "nonchilled": 1 1 1 1 1 1 1 1 1 1 ...
##  $ conc     : int  95 175 250 350 500 675 1000 95 175 250 ...
##  $ uptake   : num  16 30.4 34.8 37.2 35.3 39.2 39.7 13.6 27.3 37.1 ...
```

```r
# exttract data of Qn1 plant
Qn1 <- subset(txt, Plant == "Qn1")
# save Qn1 to .txt.
# Though the file extension is .txt, the data is actually stored in a CSV format.
write.table(Qn1, file = "data/CO2-Qn1.txt", row.names = FALSE, quote = FALSE, sep = ",")

# save Qn1 to .csv
write.csv(Qn1, file = "data/CO2-Qn1.csv", row.names = FALSE, quote = FALSE)
# use read.csv to read CO2-Qn1.csv
# fileEncoding declares the encoding used on the file, which must be careful
# when you read data containing Chinese on Windows. In such a situation, you can
# try fileEncoding = "cp936".
# stringsAsFactors is a logical value indicating whether character vectors should
# be converted to factors. The default is TRUE, while some functions doesn't work
# with factors.
```

```
csv <- read.csv(file = "data/CO2-Qn1.csv", header = TRUE, sep = ",",
                fileEncoding = "utf-8", stringsAsFactors = FALSE)
str(csv)
```

```
## 'data.frame':    7 obs. of  5 variables:
##  $ Plant    : chr  "Qn1" "Qn1" "Qn1" "Qn1" ...
##  $ Type     : chr  "Quebec" "Quebec" "Quebec" "Quebec" ...
##  $ Treatment: chr  "nonchilled" "nonchilled" "nonchilled" "nonchilled" ...
##  $ conc     : int  95 175 250 350 500 675 1000
##  $ uptake   : num  16 30.4 34.8 37.2 35.3 39.2 39.7
```

```
# install xlsx package
if(!"xlsx" %in% installed.packages()) install.packages("xlsx")
# load xlsx pacakage
library(xlsx)
```

```
## Loading required package: rJava
## Loading required package: xlsxjars
```

```
# write Qn1 to .xls
write.xlsx(Qn1, file = "data/CO2-Qn1.xls", sheetName = "Sheet1", col.names = TRUE,
           row.names = FALSE)
# use read.xlsx to read CO2-Qn1.xls
xlsx <- read.xlsx(file = "data/CO2-Qn1.xls", sheetName = "Sheet1")
str(xlsx)
```

```
## 'data.frame':    7 obs. of  5 variables:
##  $ Plant    : Factor w/ 1 level "Qn1": 1 1 1 1 1 1 1
##  $ Type     : Factor w/ 1 level "Quebec": 1 1 1 1 1 1 1
##  $ Treatment: Factor w/ 1 level "nonchilled": 1 1 1 1 1 1 1
##  $ conc     : num  95 175 250 350 500 675 1000
##  $ uptake   : num  16 30.4 34.8 37.2 35.3 39.2 39.7
```

```
# install foreign package
if(!"foreign" %in% installed.packages()) install.packages("foreign")
# load foreign pacakage
library(foreign)
# write Qn1 to .dbf
write.dbf(Qn1, file = "data/CO2-Qn1.dbf")
# use read.dbf to read CO2-Qn1.dbf
dbf <- read.dbf(file = "data/CO2-Qn1.dbf")
str(dbf)
```

```
## 'data.frame':    7 obs. of  5 variables:
##  $ Plant    : Factor w/ 1 level "Qn1": 1 1 1 1 1 1 1
##  $ Type     : Factor w/ 1 level "Quebec": 1 1 1 1 1 1 1
##  $ Treatment: Factor w/ 1 level "nonchilled": 1 1 1 1 1 1 1
##  $ conc     : int  95 175 250 350 500 675 1000
##  $ uptake   : num  16 30.4 34.8 37.2 35.3 39.2 39.7
##  - attr(*, "data_types")= chr  "C" "C" "C" "N" ...
```

```
# list objects in current environment
ls()
```

```
## [1] "CO2"  "csv"  "dbf"  "Qn1"  "txt"  "x"    "xlsx"
```

```
# save txt, csv, xlsx and dbf .RData or .rda
save(txt, csv, xlsx, dbf, file = "data/format.RData")
# remove txt, csv, xlsx and dbf from current environment
rm(txt, csv, xlsx, dbf)
ls()
```

```
## [1] "CO2" "Qn1" "x"
```

```
# load format.RData
load(file = "data/format.RData")
# list objects in current environment
ls()
```

```
## [1] "CO2"  "csv"  "dbf"  "Qn1"  "txt"  "x"    "xlsx"
```

Demostration

The `SURF_CLI_CHN_MUL_DAY-TEM-12001-201409.TXT` contains daily temporature collected from all meteorological stations across China in September 2014, which is downloaded from the dataset of **SURF_CLI_CHN_MUL_DAY** v3.0 produced by CMDSSS. The following code demostrates how to preprocess the **SURF_CLI_CHN_MUL_DAY** dataset and extract your desired data in R, which is from my own research.

```
# use read.table to read dataset in TXT
data <- read.table(file = "data/CMDSSS/SURF_CLI_CHN_MUL_DAY-TEM-12001-201409.TXT")
str(data)
```

```
## 'data.frame':    24240 obs. of  13 variables:
##  $ V1 : int  50136 50136 50136 50136 50136 50136 50136 50136 50136 50136 ...
##  $ V2 : int  5258 5258 5258 5258 5258 5258 5258 5258 5258 5258 ...
##  $ V3 : int  12231 12231 12231 12231 12231 12231 12231 12231 12231 12231 ...
##  $ V4 : int  4385 4385 4385 4385 4385 4385 4385 4385 4385 4385 ...
##  $ V5 : int  2014 2014 2014 2014 2014 2014 2014 2014 2014 2014 ...
##  $ V6 : int  9 9 9 9 9 9 9 9 9 9 ...
##  $ V7 : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ V8 : int  94 83 82 153 152 157 170 144 89 80 ...
##  $ V9 : int  196 219 144 245 264 242 296 219 166 227 ...
##  $ V10: int  29 1 4 97 85 67 57 104 47 -33 ...
##  $ V11: int  0 0 0 0 0 0 0 0 0 0 ...
##  $ V12: int  0 0 0 0 0 0 0 0 0 0 ...
##  $ V13: int  0 0 0 0 0 0 0 0 0 0 ...
```

```
# V1: station id
# V2: latitude, the last two digits are minitue and the remaining digits are degrees
# V3: longitude, the last two digits are minitue and the remaining digits are degrees
```

```r
# V4: altitude in 0.1 meter. When the station altitude is estimated rather than
#     measured, 100000 is added.
# V5: year
# V6: month
# V7: days of the month
# V8: average temperature in 0.1 degree Celsius
# V9: daily maximum temperature in 0.1 degree Celsius
# V10: daily minimum temperature in 0.1 degress Celsius.
#      When the actual temperature is higher than the higher limit of the instrument
#      measuring range, 10000 is added; When the actual temperature is lower than the
#      lower limit, 10000 is subtracted.
#      32766 indicates observations are not available (NA).
# V11: quality control code of average temperature
# V12: quality control code of daily maximum temperature
# V13: quality control code of daily minimum temperature
colnames(data) <- c("id", "lat", "lng", "alt", "year", "month", "day", "meanT",
                    "maxT", "minT", "meanQC", "maxQC", "minQC")

# convert dm (the last two digits of dm represent minitue) into degree
DM2D <- function(dm) {
  dm %/% 100 + (dm %% 100) / 60
}

# process function for temperature
process <- function(df) {
  df[, 2:3] <- DM2D(df[, 2:3]) # latitude and longitude
  df[, 4] <- ifelse(df[, 4] > 100000, df[, 4] - 100000, df[, 4]) # altitude
  # add a date-time variable for convenient date-time calculation
  df$ymd <- ISOdate(df$year, df$month, df$day)
  df[df[, 8] == 32766, 8] <- NA
  df[df[, 9] == 32766, 9] <- NA
  df[df[, 10] == 32766, 10] <- NA
  df[, 8:10] <- df[, 8:10] / 10
  df
}

data <- process(data)
str(data)
```

```
## 'data.frame':    24240 obs. of  14 variables:
##  $ id    : int  50136 50136 50136 50136 50136 50136 50136 50136 50136 50136 ...
##  $ lat   : num  53 53 53 53 53 ...
##  $ lng   : num  123 123 123 123 123 ...
##  $ alt   : num  4385 4385 4385 4385 4385 ...
##  $ year  : int  2014 2014 2014 2014 2014 2014 2014 2014 2014 2014 ...
##  $ month : int  9 9 9 9 9 9 9 9 9 9 ...
##  $ day   : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ meanT : num  9.4 8.3 8.2 15.3 15.2 15.7 17 14.4 8.9 8 ...
##  $ maxT  : num  19.6 21.9 14.4 24.5 26.4 24.2 29.6 21.9 16.6 22.7 ...
##  $ minT  : num  2.9 0.1 0.4 9.7 8.5 6.7 5.7 10.4 4.7 -3.3 ...
##  $ meanQC: int  0 0 0 0 0 0 0 0 0 0 ...
##  $ maxQC : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ minQC : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ ymd   : POSIXct, format: "2014-09-01 12:00:00" "2014-09-02 12:00:00" ...
```

```r
# extract data collected from Miyun station whose id is 54416 and discard quality control codes
x <- subset(data[, !names(data) %in% c("meanQC", "maxQC", "minQC")], id == 54416)
str(x)
```

```
## 'data.frame':    30 obs. of  11 variables:
##  $ id   : int  54416 54416 54416 54416 54416 54416 54416 54416 54416 54416 ...
##  $ lat  : num  40.4 40.4 40.4 40.4 40.4 ...
##  $ lng  : num  117 117 117 117 117 ...
##  $ alt  : num  718 718 718 718 718 718 718 718 718 718 ...
##  $ year : int  2014 2014 2014 2014 2014 2014 2014 2014 2014 2014 ...
##  $ month: int  9 9 9 9 9 9 9 9 9 9 ...
##  $ day  : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ meanT: num  22.2 19.5 20.6 22.5 22.9 23.5 23.2 21.5 20.4 20.8 ...
##  $ maxT : num  25.9 22.5 29.4 32.9 29.2 28.7 28.1 29.1 28.9 28.2 ...
##  $ minT : num  20.1 18.6 12.7 15.1 17.4 19.1 20.4 12.3 13.3 15.6 ...
##  $ ymd  : POSIXct, format: "2014-09-01 12:00:00" "2014-09-02 12:00:00" ...
```

```r
# summaries of meanT, maxT, and minT
summary(x[, c("meanT", "maxT", "minT")])
```

```
##      meanT            maxT            minT
##  Min.   :10.20   Min.   :14.60   Min.   : 6.80
##  1st Qu.:17.10   1st Qu.:22.57   1st Qu.:12.22
##  Median :19.45   Median :25.25   Median :14.85
##  Mean   :19.04   Mean   :25.04   Mean   :14.52
##  3rd Qu.:20.60   3rd Qu.:28.05   3rd Qu.:16.77
##  Max.   :23.50   Max.   :32.90   Max.   :20.40
```

```r
# save the extracted data to external file for further data analysis
save(x, file = "data/Miyun-TEM-201409.RData")
```

References

The following are materials on R data import/export that you can access on the Web.

- R Data Import/Export