Lab 03: Introduction to the R language (2)

Seoncheol Park

Instructions

- Students should solve exercises provided in the in-class exercise session every Friday.
- Students should write (a single) R code about answers to exercises and submit it on the course website.
  - for 11628 (Fri 10:30~12:00)
  - for 13300 (Fri 09:00~10:30)
- If you have any questions, please raise your hand. The professor and TA will help you.
- We will **not** give any scores for all **late submissions**. Please keep the time.
- You may leave early after submitting your answers on the course website.
- Before leaving the classroom, please **check** whether your answers are uploaded well.

Exercise A: Solar radiation observations

The following are a sample of observations on incoming solar radiation at a greenhouse.

```
11.1   10.6    6.3   8.8   10.7    11.2    8.9   12.2
```

Exercise A-1. Assign the data to an object called `solar.radiation`.

Exercise A-2. Find the mean, median, and variance of `solar.radiation`.

Exercise A-3. Add 10 to each observation of `solar.radiation`, and assign the result to `sr10`. Find the mean, median, range and variance of `sr10`. Which statistics change, and by how much?

Exercise A-4. Multiply each observation by `-2`, and assign the result to `srm2`. Find the mean, median, and variance of `srm2`. How do the statistics change now?

Exercise A-5. Plot a histogram of th `solar.radiation`, `sr10`, and `srm2`.

Exercise B: Date and times in R using `Date`, `POSIXct` and `POSIXlt`

- Note that R has numerous functions to manipulate times and dates. There are three basic date and time classes: `Date`, `POSIXct` and `POSIXlt`.
  - Class `Date` handles dates **without times**.
  - `POSIXct` stores date and time in seconds with the number of seconds beginning at 1 January 1970. Negative numbers are used to store dates prior to 1970. Thus, the `POSIXct` format stores each date and time a single value in units of seconds.
  - `POSIXlt` stores dates and times as a list of components: second, minute, hour, day, month, year, time zone etc.
- **Data Tip:** The `unclass` method in R allows you to view how a particular R object is stored.
- Codes for time zones (`tz`) can be also found in this time zone table.

```
date_1970 <- "1970-01-02 09:00:01" #Seoul: UCT + 9

datetime1 <- as.POSIXct(date_1970, tz="Asia/Seoul")
datetime1
```

```
[1] "1970-01-02 09:00:01 KST"
```

```
#unclass(datetime1) #POSIXct -> numeric

#datetime2 <- as.POSIXlt(date_1970, tz="Asia/Seoul") #try it!
#datetime2
#unclass(datetime2) #POSIXct -> list

#datetime3 <- as.Date(date_1970, tz="Asia/Seoul") #try it!
#datetime3  #try it!
```

`as.Date()`, `as.POSIXct()`, `as.POSIXlt()`

- `as.Date()`, `as.POSIXct()`and `as.POSIXlt()`convert dates and times in character forms to classes of dates and times.
- `as.Date()`, `as.POSIXct()`and `as.POSIXlt()`accept various input formats.

The default input formats are

1. **year-month-day hour:minutes:seconds** or
2. **year/month/day hour:minutes:seconds.**

```
as.Date("2019/01/14 14:17:30")
```

```
[1] "2019-01-14"
```

```
#as.POSIXct("2019/01/14 14:17:30") #try it!
#as.POSIXlt("2019/01/14 14:17:30") #try it!
```

- If the input format is not standard, we need to set the `format` argument to map the displayed format.
  - %b abbreviated month name
  - %m month as decimal number (01–12)
  - %c date and time
  - %d day of the month as decimal number (01–31)
  - %e day of the month as decimal number (1–31)
  - %H hours as decimal number (00–23); strings such as 24:00:00 are accepted for input
  - %I hours as decimal number (01–12)
  - %M minute as decimal number (00–59)
  - %S second as integer (00–61)
  - %OS seconds including fractional seconds
  - %Y year with century
  - %y year without century (00–99)
- The full list of allowed formats can be found by `?strptime()`.

```
as.Date("14jan2019 14:17:30", "%d%b%Y")
```

```
[1] "2019-01-14"
```

```
#as.POSIXct("14jan2019 14:17:30", format = "%d%b%Y %H:%M:%S")  #try it!
#as.POSIXlt("14jan2019 14:17:30", format = "%d%b%Y %H:%M:%S")  #try it!
#as.Date("14/01/2019T14:17:30", "%d/%m/%Y")  #try it!
#as.POSIXct("14/01/2019T14:17:30", format = "%d/%m/%YT%H:%M:%S", tz = "GMT")  #try it!
#as.POSIXlt("14/01/2019T14:17:30", format = "%d/%m/%YT%H:%M:%S", tz = "GMT")  #try it!
```

Exercise B-1. Use all of `as.Date()`, `as.POSIXct()`, `as.POSIXlt()`functions to parse each of the following dates:

```
d1 <- "06-Jun-2017 15:14:46"
d2 <- "12/30/14T19:23"
```

Your answers should be `Date`, `POSIXct`, and `POSIXlt` object. If you parse correctly, answers in the Console Pane may be

```
[1] "2017-06-06"

[1] "2017-06-06 15:14:46 KST"

[1] "2017-06-06 15:14:46 KST"

[1] "2014-12-30"

[1] "2014-12-30 19:23:00 KST"

[1] "2014-12-30 19:23:00 KST"
```

Exercise C: Date and times in R using `lubridate` package

- `lubridate` provides more intuitive ways to convert characters to dates and times.
- You can install and call all functions in the `lubridate` package via

```
install.packages("lubridate")
library(lubridate)
```

- `ymd()`, `ydm()`, `mdy()`, `myd()`, `dmy()`, `dym()`, `yq()`parses dates with year, month, and day components.

```
ymd("2019/01/14")
```

```
[1] "2019-01-14"
```

```
#ydm("2019-14-01") #try it!
#dmy("14jan2019") #try it!
```

- `hm()`, `ms()`, `hms()`parses periods with hour, minute, and second components.

```
hms("14:17:30")
```

```
[1] "14H 17M 30S"
```

```
#hm("14:17") #try it!
#ms("17:30") #try it!
```

- `ymd_hms()`, `ymd_hm()`, `ymd_h()`, `dmy_hms()`, `dmy_hm()`, `dmy_h()`, `mdy_hms()`, `mdy_hm()`, `mdy_h()`, `ydm_hms()`, `ydm_hm()`, `ydm_h()`parses date-times with year, month, and day, hour, minute, and second components.

```
ymd_hms("2019/01/14 14:17:30")
```

```
[1] "2019-01-14 14:17:30 UTC"
```

```
#dmy_hms("14jan2019 14:17:30") #try it!
#dmy_hms("14/01/2019T14:17:30") #try it!
```

Exercise C-1. Use the appropriate `lubridate` function to parse each of the following dates:

```
e1 <- "January 1, 2023 15:14:46"
e2 <- c("August 19 (2005)", "July 1 (2005)")
```

If you parse correctly, answers in the Console Pane may be

```
[1] "2023-01-01 15:14:46 KST"
```

```
[1] "2005-08-19 KST" "2005-07-01 KST"
```

Exercise D: Calculations with dates and times

`difftime()`

```
#strptime: character -> POSIXlt
x <- strptime("2019-01-14 14:17:30", "%Y-%m-%d %H:%M:%S")
y <- strptime("2018-12-14 18:10:12", "%Y-%m-%d %H:%M:%S")
x - y
```

```
Time difference of 30.8384 days
```

- The base R function `difftime()`calculates a difference of two date-time objects and returns a difftime object.

```
difftime(x, y)
```

```
Time difference of 30.8384 days
```

```
difftime(x, y, units = "hours")
```

```
Time difference of 740.1217 hours
```

```
#difftime(x, y, units = "mins") #try it!
#difftime(x, y, units = "secs") #try it!
#difftime(x, y, units = "days") #try it!
#difftime(x, y, units = "weeks") #try it!
```

- `difftime` objects can be converted to numeric objects with `as.numeric()`.

```
z <- difftime(x, y)
as.numeric(z, unit = "hours")
```

```
[1] 740.1217
```

```
#as.numeric(z, unit = "mins") #try it!
```

Exercise D-1. Write a function `dayspassed` that given a birthday as a `Date` object, returns how many days are passed from the given birthday. You can use `today()`function to get the current day and time.

```
dayspassed <- function(aday) {
  #assume that aday is always a `Date` object

  #fill in the blank
}
```

- If you make the function `dayspassed` correctly, answers in the Console Pane may be

```
his_birthday <- as.Date("2019/01/14")
dayspassed(his_birthday)
```

```
Time difference of 1710 days
```