Session Seven - Working with Dates and Time

By Hellen Gakuruh

June 2, 2016

Table of Contents

Session Goal

The main goal of this session is to get you started with date and time objects which will be significant during data analysis.

What we shall cover

By the end of this session you should:

- Be able to know what format, time zones, and origin mean in relation to date and time
- Know how to create date and time objects with both POSIXct and POSIXlt classes
- Be able to create date and time factor objects
- Skilled in computing and manipulating date-time obejcts

Prerequisite

To appreciate this session, you must be conversant with:

- Making function calls
- Concept of object oriented programming in R particularly classes and methods

Recognising date-time objects in R

As with every analytical program, date-time data have to be declared explicitly for the program to recognize it and treat as such. Internally, most programs store dates as numerical values from an arbitrary original date, for example "1970-01-01" for R, "0000-01-01" for Matlab, "1900-01-01" for excel and windows, hence date can be a numerical value.

R provides a number of ways of declaring or converting data into data and time. But first let's see current date and time according to R

Current date

In base R "Sys.Date()" can be used to get the current date.

```
Sys.Date()
## [1] "2016-06-08"

# Class of output
class(Sys.Date())
## [1] "Date"
```

Sys.Date produces an output of class "Date", and its format is year (four digits), hyphen, month (two digits), hyphen, then date (two digits). This format can be changed to any other date format using "format" function.

Format

Format is a generic function meaning it can be used by different data objects not just date objects. Therefore it is important to give it an object of class "Date" and include a format method.

A format method is a character vector that details how date will be presented. These methods are documented by "strptime" function under the "format" section. Default method is "%Y-%m-%d %H:%M:%S for an object with a time component and simply "%Y-%m-%d" if it does not have a time component.

As detailed by "strptime" function, formats are platform-specif but most would recognize some of the format given otherwise you may need to set LC_TIME category to an appropriate type through "Sys.setlocale" function. Notable effects are to months names and AM/PM.

```
# Platform used
.Platform$OS.type
## [1] "windows"
.Platform$GUI
## [1] "RTerm"
Sys.setlocale()
## [1] "LC_COLLATE=English_United States.1252;LC_CTYPE=English_United
States.1252;LC_MONETARY=English_United
States.1252;LC_NUMERIC=C;LC_TIME=English_United States.1252"
```

Conversion of specification is done through the symbol "%" followed by a single letter. Some of the conversion specifications include

```
# Abbreviated weekday name
format(Sys.Date(), format = "%a")
## [1] "Wed"
# Full weekday name
format(Sys.Date(), format = "%A")
## [1] "Wednesday"
# Abbreviated month name
format(Sys.Date(), format = "%b")
## [1] "Jun"
# Full month name
format(Sys.Date(), format = "%B")
## [1] "June"
# Day of the month as a decimal number
format(Sys.Date(), format = "%d")
## [1] "08"
# Conversion to m/d/y * not all on systems
format(Sys.Date(), format = "%D")
## [1] "06/08/16"
# Week of the year
format(Sys.Date(), format = "%W")
## [1] "23"
```

Do take note, these conversion methods can be used to manipulate date objects which will become quite handy during data analysis section in level three.

Current Time

To establish current time, "Sys.time()" from base R is appropriate.

```
Sys.time()
## [1] "2016-06-08 22:34:17 EAT"
```

Notice this function gives a more comprehensive output as it contains both date and time components. The characters at the end, right after time is R's version of your current time zone.

Just like date object, time can be converted to a different format depending on a given format specification or method.

```
# Current time
Sys.time()
## [1] "2016-06-08 22:34:17 EAT"
# Converting to a different format
format(Sys.time(), "%c")
## [1] "Wed Jun  8 22:34:17 2016"
# Getting the hour part
format(Sys.time(), "%H")
## [1] "22"
# Time in the 12-hour clock system
format(Sys.time(), "%r")
## [1] "10:34:17 PM"
```

Sys.time() outputs an object of class "POSIXct" and "POSIXt". We will discuss these two classes as we create some date-time objects.

Creating date-time objects in base R

To be able to do any analysis dealing with date and or time in R, you must have an object with a date-time class. A class as discussed in object oriented programming in R is important in selecting appropriate functions for data-time objects.

Therefore the date-time objects we create would have classes that are recognized by R as date and time classes. These classes fall under the POSIX system.

POSIX

Posix time also referred to as "Unix time" or "Epoch time" is a system that describes time in seconds from some origin which is 1st January 1970 in UTC time zone. Days and dates are calculated as number of seconds that have passed since this origin, this would be in numerical form like 1.465414510^{9} hence not comprehensible to humans (just like binary code).

There are two date-time classes in base R, these are "POSIXct" and "POSIXlt". A virtual class "POSIXt" exists for allowing operations on both POSIXct and POSIXlt.

```
# Methods (functions) avaliable for these classes
methods(class = "POSIXlt")[1:28];
##  [1] "[.POSIXlt"                      "[<-.POSIXlt"
##  [3] "anyNA.POSIXlt"                  "as.data.frame.POSIXlt"
##  [5] "as.Date.POSIXlt"                "as.double.POSIXlt"
##  [7] "as.matrix.POSIXlt"              "as.POSIXct.POSIXlt"
##  [9] "c.POSIXlt"                      "coerce,oldClass,S3-method"
## [11] "duplicated.POSIXlt"             "format.POSIXlt"
## [13] "initialize,oldClass-method"     "is.na.POSIXlt"
## [15] "length.POSIXlt"                 "mean.POSIXlt"
## [17] "names.POSIXlt"                  "names<-.POSIXlt"
## [19] "print.POSIXlt"                  "rep.POSIXlt"
## [21] "show,oldClass-method"           "slotsFromS3,oldClass-method"
## [23] "sort.POSIXlt"                   "summary.POSIXlt"
## [25] "Summary.POSIXlt"                "unique.POSIXlt"
## [27] "weighted.mean.POSIXlt"          "xtfrm.POSIXlt"
methods(class = "POSIXct")[1:21]
##  [1] "[.POSIXct"                      "[[.POSIXct"
##  [3] "[<-.POSIXct"                    "as.data.frame.POSIXct"
##  [5] "as.Date.POSIXct"                "as.list.POSIXct"
##  [7] "as.POSIXlt.POSIXct"             "c.POSIXct"
##  [9] "coerce,oldClass,S3-method"      "format.POSIXct"
## [11] "initialize,oldClass-method"     "mean.POSIXct"
## [13] "print.POSIXct"                  "rep.POSIXct"
## [15] "show,oldClass-method"           "slotsFromS3,oldClass-method"
## [17] "split.POSIXct"                  "summary.POSIXct"
## [19] "Summary.POSIXct"                "weighted.mean.POSIXct"
## [21] "xtfrm.POSIXct"
```

POSIXct

This class is most suitable for date/time variables in a data frame. An object of class POSIXct is a numeric vector of seconds since 1970 (R's version of original time).

POSIXlt

An object of class POSIXlt is a list with about eleven vectors, these are; time in seconds, minutes and hour, day of the month, month, year since 1900, day of the week, day of the year, daylight saving time, time zone, and offset in seconds from GMT. The latter two are platform specific.

Objects of this class are convenient for date-time manipulation.

Conversion of POSIXlt to POSIXct is easily done, however, POSIXct objects may need a time zone attribute to convert them to POSIXlt. Other date objects like those from other packages like "chron" and "dates" can also be converted to either POSIXct, POSIXlt or both. Both POSIXct and POSIXlt inherit POSIXt a virtual class.

Creating date-time objects from character and numeric vectors is done with "as.POSIXct() or POSIXlt()" or with "strptime()".

Creating POSIXct objects

Given character vectors

```
# Character vectors of dates and times
dates <- c("Aug 1, 1972", "Jan 1, 1973", "Jan 1, 1974", "Jan 1, 1975")
times <- c("23:59:59", "00:00:01", "12:02:19", "12:39:50")
datesTimes <- paste(dates, times)
datesTimes
## [1] "Aug 1, 1972 23:59:59" "Jan 1, 1973 00:00:01" "Jan 1, 1974 12:02:19"
## [4] "Jan 1, 1975 12:39:50"
class(datesTimes)
## [1] "character"
```

Creating POSIXct object is done with "as.POSIXct" function

```
# Converting character vector with dates
dateObject1 <- as.POSIXct(dates, tz = "GMT", format = "%b %e, %Y")
dateObject1
## [1] "1972-08-01 GMT" "1973-01-01 GMT" "1974-01-01 GMT" "1975-01-01 GMT"
class(dateObject1)
## [1] "POSIXct" "POSIXt"

# Converting times vector
timeObject1 <- as.POSIXct(times, tz = "GMT", format = "%H:%M:%S")
timeObject1
## [1] "2016-06-08 23:59:59 GMT" "2016-06-08 00:00:01 GMT"
## [3] "2016-06-08 12:02:19 GMT" "2016-06-08 12:39:50 GMT"
class(timeObject1)
## [1] "POSIXct" "POSIXt"
```

For the last object involving time without dates, R assigned current date to each time, if this is not what you had in mind, then include the correct dates otherwise R considers time as part of a specific day.

We can use the combined date and time to create a POSIXct object but amending argument format.

```
dateTimeObject1 <- as.POSIXct(datesTimes, format = "%b %e, %Y %H:%M:%S")
dateTimeObject1
## [1] "1972-08-01 23:59:59 EAT" "1973-01-01 00:00:01 EAT"
## [3] "1974-01-01 12:02:19 EAT" "1975-01-01 12:39:50 EAT"
class(dateTimeObject1)
## [1] "POSIXct" "POSIXt"
```

Creating POSIXlt Objects

To create a POSIXlt object, we use "as.POSIXlt" function.

```
dateTimeObject2 <- as.POSIXlt(datesTimes, format = "%b %e, %Y %H:%M:%S")
dateTimeObject2
## [1] "1972-08-01 23:59:59 EAT" "1973-01-01 00:00:01 EAT"
## [3] "1974-01-01 12:02:19 EAT" "1975-01-01 12:39:50 EAT"
class(dateTimeObject2)
## [1] "POSIXlt" "POSIXt"
```

When creating date-time objects, one consideration to make is "time zone". Time zone is system and locale specif, hence you must ensure it is correct for your current setting. Use "Sys.timezone() to see your current timezone and can check "OlsonNames()".

Creating date-time Object using "strptime()"

"strptime()" creates POSIXlt objects just like "as.POSIXlt()"

```
dateObject2 <- strptime(dates, format = "%b %e, %Y")
dateObject2
## [1] "1972-08-01 EAT" "1973-01-01 EAT" "1974-01-01 EAT" "1975-01-01 EAT"
class(dateObject2)
## [1] "POSIXlt" "POSIXt"

#
timeObject2 <- strptime(times, format = "%H:%M:%S")
timeObject2
## [1] "2016-06-08 23:59:59 EAT" "2016-06-08 00:00:01 EAT"
## [3] "2016-06-08 12:02:19 EAT" "2016-06-08 12:39:50 EAT"
class(timeObject2)
## [1] "POSIXlt" "POSIXt"

#
dateTimeObject3 <- strptime(datesTimes, format = "%b %e, %Y %H:%M:%S")
dateTimeObject3
## [1] "1972-08-01 23:59:59 EAT" "1973-01-01 00:00:01 EAT"
## [3] "1974-01-01 12:02:19 EAT" "1975-01-01 12:39:50 EAT"
class(dateTimeObject3)
## [1] "POSIXlt" "POSIXt"
```

Converting numeric data to date and time object

Numerical values can be converted to either POSIXct or POSIXlt object by specifying origin (Julian time) or the beginning of time. Sometimes this might be a better option when importing data into R, as characters need to be formatted.

```
numPOSIXct <- as.POSIXct(1938494958, origin = "1900-01-01")
numPOSIXct; class(numPOSIXct)
## [1] "1961-06-06 09:49:18 EAT"
## [1] "POSIXct" "POSIXt"
numPOSIXlt <- as.POSIXlt(3859583882, origin = "1970-01-01")
numPOSIXlt; class(numPOSIXlt)
```

```
## [1] "2092-04-21 05:38:02 EAT"
## [1] "POSIXlt" "POSIXt"
```

Creating date and time factor vectors

Date and time objects can be converted to factor vectors with "cut" function. The "breaks" argument is used to specify the number of levels the vector will have.

```
# Leap vector as a factor vector with years as levels
dateFactor1 <- cut(.leap.seconds, breaks = "year")
# Leap vector converted to a two level factor vector
dateFactor2 <- cut(.leap.seconds, breaks = 2)
# Leap vector as binary ordinal (ordered) vector
dateFactor3 <- cut(.leap.seconds, breaks = c(.leap.seconds[1],
.leap.seconds[14], .leap.seconds[26]), labels = c("Before 1990's", "After
1990s"), include.lowest = TRUE, right = TRUE, ordered_result = TRUE)
```

Creating Date objects

Date objects (without time) can be created with "as.Date" function.

```
pureDateObject <- as.Date(dates, format = "%b %e, %Y")
pureDateObject
## [1] "1972-08-01" "1973-01-01" "1974-01-01" "1975-01-01"
class(pureDateObject)
## [1] "Date"
```

Computation with date and time objects

Date and time objects can be used to carry out computation such as addition and subtraction of seconds and obtaining differences between two time objects.

```
# Difference between two date objects
begining <- as.Date(17084, "1970-01-01")
end <- Sys.Date()
end - begining
## Time difference of -124 days
```

We can create a sequence of dates for computation with "seq" function

```
# Weekly spaced sequence of dates
seq(from = begining, to = end, by = -7)
##  [1] "2016-10-10" "2016-10-03" "2016-09-26" "2016-09-19" "2016-09-12"
##  [6] "2016-09-05" "2016-08-29" "2016-08-22" "2016-08-15" "2016-08-08"
## [11] "2016-08-01" "2016-07-25" "2016-07-18" "2016-07-11" "2016-07-04"
## [16] "2016-06-27" "2016-06-20" "2016-06-13"
```

Addition between two date objects is not possible.

```
as.Date(29482, "1900-01-01") + Sys.Date()
## Error in `+.Date`(as.Date(29482, "1900-01-01"), Sys.Date()) :
##  binary + is not defined for "Dates" objects
```

Computation with numeric value is feasible

```
# Today's date
Sys.Date()
## [1] "2016-06-08"
# Date 10 days ago
Sys.Date() - 10
## [1] "2016-05-29"
# Date 10 days from today
Sys.Date() + 10
## [1] "2016-06-18"

# Eight days ago computed with number of seconds
Sys.time() - (24*60*60)*8
## [1] "2016-05-31 22:34:19 EAT"
```

Logical operators can also be used to manipulate date-time objects.

```
.leap.seconds[11] > .leap.seconds[8]
## [1] TRUE
.leap.seconds <= .leap.seconds[15]
##  [1]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
## [12]  TRUE  TRUE  TRUE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [23] FALSE FALSE FALSE FALSE
.leap.seconds[.leap.seconds >= .leap.seconds[22]]
## [1] "1999-01-01 03:00:00 EAT" "2006-01-01 03:00:00 EAT"
## [3] "2009-01-01 03:00:00 EAT" "2012-07-01 03:00:00 EAT"
## [5] "2015-07-01 03:00:00 EAT"
```

Extracting parts of a date-time object

Specific parts of a date object can be extracted with some extractor function.

```
# Date-Time Objects
dateTimeObject1
## [1] "1972-08-01 23:59:59 EAT" "1973-01-01 00:00:01 EAT"
## [3] "1974-01-01 12:02:19 EAT" "1975-01-01 12:39:50 EAT"
dateTimeObject2
## [1] "1972-08-01 23:59:59 EAT" "1973-01-01 00:00:01 EAT"
## [3] "1974-01-01 12:02:19 EAT" "1975-01-01 12:39:50 EAT"
# Classes of the objects
class(dateTimeObject1)
## [1] "POSIXct" "POSIXt"
class(dateTimeObject2)
## [1] "POSIXlt" "POSIXt"
# Day of the week
weekdays(dateTimeObject1)
## [1] "Tuesday"   "Monday"    "Tuesday"   "Wednesday"
weekdays(dateTimeObject2)
## [1] "Tuesday"   "Monday"    "Tuesday"   "Wednesday"
# Month
```

```
months(dateTimeObject1)
## [1] "August"  "January" "January" "January"
months(dateTimeObject2)
## [1] "August"  "January" "January" "January"
# Quarter (Q1:Q4)
quarters(dateTimeObject1)
## [1] "Q3" "Q1" "Q1" "Q1"
quarters(dateTimeObject2)
## [1] "Q3" "Q1" "Q1" "Q1"
# Number of days since origin (default is 1970-01-01)
julian(dateTimeObject1)
## Time differences in days
## [1]  943.875 1095.875 1461.377 1826.403
## attr(,"origin")
## [1] "1970-01-01 GMT"
julian(dateTimeObject2, origin = "1900-01-01")
## Time differences in days
## [1] 26510.98 26662.98 27028.48 27393.50
## attr(,"origin")
## [1] "1900-01-01 LMT"
```

Gauge yourself

Do you have the expected tools and skills from this session?

1. What is the default date format in R? Write in character form and in format method using "%"
2. Write a code to convert "2016-06-08" into this format "Wed Jun 08"
3. What went wrong with these conversion? a. format(Sys.Date(), "C") a. format(Sys.time(), %H)
4. Get your current time in the 12-hour clock system
5. Extract the years from ".leap.seconds" object
6. Convert "2nd June 2016" to POSIXct object
7. Convert "9th June 2016 21:39:40" to POSIXlt object
8. Create a factor vector from seq(from = as.Date("2015-01-01"), to = as.Date("2015-12-31"), by = 1) with four levels labeled "Q1", "Q2", "Q3", "Q4"
9. Why would would this computation not work as.Date("2016-06-08") + as.Date("2015-06-08")
10. Can numeric values be converted to date-time object? If so, what is of importance in conversion.

You are ready for the seventh session if you know these

1. Default date format is "Year-month-date", format method is "%Y-%m-%d" or "%F" (ISO 8601 format)
2. This comes from R documentation, "2016-06-08" can be converted to "Wed Jun 08" with format("2016-06-08", "%a %b %d")

3. The problem with "format(Sys.Date(), "C")" is that "%" is missing in the last part of "C". As for "format(Sys.time(), %H)", format is not a character vector, hence R will see as an object of which it is not.
4. To get 12-hour clock version of current time, "format(Sys.time(), format = "%r")"
5. Leap years can be extracted from ".leap.seconds" object with "format(.leap.seconds, "%Y")"
6. "2nd June 2016" can be converted to POSIXct object by as.POSIXct("2nd June 2016", format = "%dnd %B %Y")
7. Conversion of "9th June 2016 21:39:40" to POSIXlt can be done with as.POSIXlt("9th June 2016 21:39:40", tz = "UTC", "%dth %B %Y %T")
8. Cut function can be used to convert date-time object to a factor vector. In this case the code would be cut(x = seq(as.Date("2015-01-01"), as.Date("2015-12-31"), by = 1), breaks = 4, labels = c("Q1", "Q2", "Q3", "Q4"))
9. Addition of two date-time objects is not possible as there are not methods written for the, specifically they is no method for "+"
10. Yes, numeric values can be converted to date-time objects as long as origin is specified.

References

Quick reference on functions

1. ?Sys.time
2. ?Sys.Date
3. ?format
4. ?"date-time"
5. ?as.POSIXct
6. ?as.POSIXlt
7. ?as.strptime
8. ?as.Date
9. ?Julian
10. ?cut

Other Packages on date and time

1. chron
2. POSIXct
3. date