Give R a date

Emmanuel Kemel

December 11, 2016

This note presents how to manipulate date data with R.

1 Date objects

- In R, there are different classes of objects for storing data (e.g factors, data.frame, list). There is also a class ”Date” that is suitable to store date data.
- When importing a data set, dates will be likely to be considered as character strings.
- What you have to do is to convert these strings into date objects. Because dates can be specified in different formats (ex: Thursday December 8th , 08/11/2016, 8 December 2016) you have to precise the format of your date data. The conversion is made by function as.Date() that takes a arguments the character string that needs to be converted, and a description of the format of the date.

```
mybirthday_string="28/06/1984"
mybirthday_date=as.Date(mybirthday_string, "%d/%m/%Y")
class(mybirthday_string)

## [1] "character"

class(mybirthday_date)

## [1] "Date"
```

The argument that describes the format of the date is a character string that describes each component of the date. A complete list of the possible components of the date is given in Table 1. Note that the sign used to separate the components of the date must be reported ”as is”. This comment applies for empty spaces also.

```
my_date=as.Date("08 12 2016", "%d %m %Y")
```

The format used by defaults is ”2000-12-31”. In this case, the format needs not to be precised.

```
test=as.Date("2000-12-31")
class(test)

## [1] "Date"
```

The previous examples deal with vectors of length=1, but the function can be used with larger vectors.

```
my_dates=as.Date(c("08 12 2016","09 12 2017"), "%d %m %Y")
length(my_dates)

## [1] 2
```

If you want to create a date object manually, you can use function ISOdate(2000,1,1) that creates a date object from the standard English system: year, month , day.

2 Useful functions

The interest of defining date objects as objects with class Date is that several functions are specific to these objects.

2.1 Get the weekday corresponding to a given date

Function **weekdays()** returns the day of the week corresponding to given date. If your parents do not remember which day was your birth day, ask R. More seriously, this function can be useful to identify ”specific day” such as Sundays in time series.

```
weekdays(mybirthday_date)

## [1] "Thursday"
```

2.2 Creating time sequences

seq() is generally used to create sequences of numeric values, but it also applies to dates. The functions takes a starting dates, a ”by” arguement that precises the step (”sec”, ”min”, ”hour”, ”day”, ”DSTday”, ”week”, ”month”, ”quarter” or ”year”), possibly multiplied by an integer, and the end date or the length of the output sequence.

```
seq(mybirthday_date, length.out=10, by="2 years")

##  [1] "1984-06-28" "1986-06-28" "1988-06-28" "1990-06-28" "1992-06-28"
##  [6] "1994-06-28" "1996-06-28" "1998-06-28" "2000-06-28" "2002-06-28"

seq(as.Date("1910/1/1"), as.Date("1919/1/1"), "years")

##  [1] "1910-01-01" "1911-01-01" "1912-01-01" "1913-01-01" "1914-01-01"
##  [6] "1915-01-01" "1916-01-01" "1917-01-01" "1918-01-01" "1919-01-01"

seq(as.Date("1910/1/1"), as.Date("1910/6/1"), "weeks")

##  [1] "1910-01-01" "1910-01-08" "1910-01-15" "1910-01-22" "1910-01-29"
##  [6] "1910-02-05" "1910-02-12" "1910-02-19" "1910-02-26" "1910-03-05"
## [11] "1910-03-12" "1910-03-19" "1910-03-26" "1910-04-02" "1910-04-09"
## [16] "1910-04-16" "1910-04-23" "1910-04-30" "1910-05-07" "1910-05-14"
## [21] "1910-05-21" "1910-05-28"
```

2.3 Rounding dates

cut() is a general function that originaly applies to numerical values, but can also apply to the Date class. In these cases, it rounds the date to a larger time unit. Rounding a specificy day to the related month could be done easily without this function, but things would become more complicated if a specific day had to be turned into the week of the year it belongs to. Function cut() runs theseconvertions easily.

```
some_sequence=seq(as.Date("1910/1/1"), as.Date("1911/1/1"), "2 weeks")
cut(some_sequence, "quarter")

##  [1] 1910-01-01 1910-01-01 1910-01-01 1910-01-01 1910-01-01 1910-01-01
##  [7] 1910-01-01 1910-04-01 1910-04-01 1910-04-01 1910-04-01 1910-04-01
## [13] 1910-04-01 1910-07-01 1910-07-01 1910-07-01 1910-07-01 1910-07-01
## [19] 1910-07-01 1910-07-01 1910-10-01 1910-10-01 1910-10-01 1910-10-01
## [25] 1910-10-01 1910-10-01 1910-10-01
## Levels: 1910-01-01 1910-04-01 1910-07-01 1910-10-01
```

```
t=cut(some_sequence, "quarter")
class(t)
```

```
## [1] "factor"
```

```
t=as.Date(t)
weekdays(t)
```

```
##  [1] "Saturday" "Saturday" "Saturday" "Saturday" "Saturday" "Saturday"
##  [7] "Saturday" "Friday"   "Friday"   "Friday"   "Friday"   "Friday"
## [13] "Friday"   "Friday"   "Friday"   "Friday"   "Friday"   "Friday"
## [19] "Friday"   "Friday"   "Saturday" "Saturday" "Saturday" "Saturday"
## [25] "Saturday" "Saturday" "Saturday"
```

Note that cut() returns a factor. This factor may have to be reconverted to a date for its next use. Note also that the levels of the created factor correspond to the first day of the specified unit.

2.4 Addition or substractions of dates

Function difftime() takes two dates as arguments and returns the time differences between the two dates, expressed in a specified unit. Note that difftime() returns objects of class difftime, but these objects can easility be converted to numericals with as.numeric().

Note that the time differences can be expressed on finer units than a day. In fact, the class date is a particular case of time objects, that allow to handle any time value with a precision of one second (see Sect. 3)

```
date1=as.Date("08 12 2016", "%d %m %Y")
date2=as.Date("08 11 2017", "%d %m %Y")
difftime(date2,date1)
```

```
## Time difference of 335 days
```

```
difftime(date2,date1, units="weeks")
```

```
## Time difference of 47.85714 weeks
```

```
difftime(date2,date1, units="hours")
```

```
## Time difference of 8040 hours
```

```
difftime(date2,date1, units="secs")
```

```
## Time difference of 28944000 secs
```

```
difftime(date2,date1, units="secs")+1
```

```
## Time difference of 28944001 secs
```

```
as.numeric(difftime(date2,date1, units="secs"))
```

```
## [1] 28944000
```

Note that numeric values can be added to dates.

```
date1=as.Date("08 12 2016", "%d %m %Y")+1
print(date1)

## [1] "2016-12-09"
```

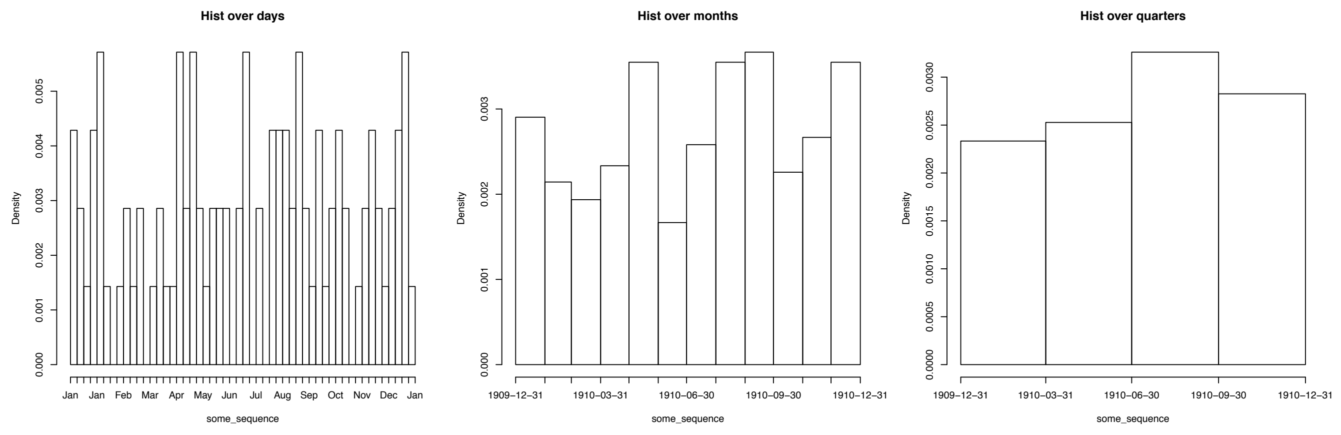
The numerical values added to a date object will be rounded to an integer and considered as days. Note that this is not the case with more general time classes (see Sect) where added or substracted numerical values are assumed to be seconds.

2.5 Plots with dates

Suppose that you want to plot the histogram of date values. You may want to change the units of the bars: one bar per day, or one bar per month... The unit can be specified with option breaks.

In order to illustrate this, the next chunk generates a random sequence of days over year 1910. The sequence is then plotted.

```
some_sequence=as.Date("1910/1/1")+364*runif(100)
par(mfrow=c(1,3))
hist(some_sequence, breaks="weeks", main="Hist over days")
hist(some_sequence, breaks="months", main="Hist over months")
hist(some_sequence, breaks="quarters", main="Hist over quarters")
```



3 Precisions about the date/time format

Strictly speaking, Date object are of class "POSIX1t", which is one of the two formats that R uses to handle time values. The other one is POSIXct. Therefore, the most general way to manipulate time is to specify explicitly the class that needs to be used. Check help(DateTimeClasses) for complete descriptions of classes available to handle date and time.

Abreviation	Description
%a	Abbreviated weekday name
%A	Full weekday name
%b	Abbreviated month name
%B	Full month name in the current locale
%c	Date and time. Locale-specific on output, ”%a %b %e %H:%M:%S %Y” on input.
%C	Century (00-99): the integer part of the year divided by 100.
%d	Day of the month as decimal number (01-31).
%D	Date format such as %m/%d/%y: the C99 standard says it should be that exact format
%e	Day of the month as decimal number (1-31)
%F	Equivalent to %Y-%m-%d (the ISO 8601 date format).
%g	The last two digits of the week-based year (see %V)
%G	The week-based year (see %V) as a decimal number
%h	Equivalent to %b.
%H	Hours as decimal number (00-23) (strings such as 24:00:00 are accepted for input)
%I	Hours as decimal number (01-12).
%j	Day of year as decimal number (001-366).
%m	Month as decimal number (01-12).
%M	Minute as decimal number (00-59).
%n	Newline on output, arbitrary whitespace on input.
%p	AM/PM indicator in the locale. Used in conjunction with %I and not with %H.
%r	The 12-hour clock time (using the locale’s AM or PM).
%R	Equivalent to %H:%M.
%S	Second as integer (00-61)
%t	Tab on output, arbitrary whitespace on input.
%T	Equivalent to %H:%M:%S.
%u	Weekday as a decimal number (1-7, Monday is 1).
%U	Week of the year as decimal number (00-53) using Sunday as the first day 1 of the week.
%V	Week of the year as decimal number (01-53)
%w	Weekday as decimal number (0-6, Sunday is 0).
%W	Week of the year as decimal number (00-53) using Monday as the first day of week.
%x	Date. Locale-specific on output, ”%y/%m/%d” on input.
%X	Time. Locale-specific on output, ”%H:%M:%S” on input.
%y	Year without century (00-99)
%Y	Year with century
%z	Signed offset in hours and minutes from UTC
%Z	(Output only.) Time zone abbreviation as a character string (empty if not available)

Table 1: List of possible components of a date

Annex

The table lists the different codes that must be used to specify the format of a date