Dates and Times in R

The measurement of time is highly idiosyncratic. Successive years start on different days of the week. There are months with different numbers of days. Leap years have an extra day in February. Americans and Britons put the day and the month in different places: 3/4/2006 is March 4 for the former and April 3 for the latter. Occasional years have an additional 'leap second' added to them because friction from the tides is slowing down the rotation of the earth from when the standard time was set on the basis of the tropical year in 1900. The cumulative effect of having set the atomic clock too slow accounts for the continual need to insert leap seconds (32 of them since 1958). There is currently a debate about abandoning leap seconds and introducing a 'leap minute' every century or so instead. Calculations involving times are complicated by the operation of time zones and daylight saving schemes in different countries. All these things mean that working with dates and times is excruciatingly complicated. Fortunately, R has a robust system for dealing with this complexity. To see how R handles dates and times, have a look at Sys.time():

```
Sys.time()
```

```
[1] "2005-10-23 10:17:42 GMT Daylight Time"
```

The answer is strictly hierarchical from left to right: the longest time scale (years) comes first, then month then day separated by hyphens (minus signs), then there is a blank space and the time, hours first (in the 24-hour clock) then minutes, then seconds separated by colons. Finally there is a character string explaining the time zone. You can extract the date from Sys.time() using substr like this:

```
substr(as.character(Sys.time()),1,10)
```

```
[1] "2005-10-23"
```

or the time

```
substr(as.character(Sys.time()),12,19)
```

```
[1] "10:17:42"
```

If you type

```
unclass(Sys.time())
```

```
[1] 1130679208
```

you get the number of seconds since 1 January 1970. There are two basic classes of date/times. Class POSIXct represents the (signed) number of seconds since the beginning of 1970 as a numeric vector: this is more convenient for including in dataframes. Class POSIXlt is a named list of vectors closer to human-readable forms, representing seconds, minutes, hours, days, months and years. R will tell you the date and time with the date function:

```
date()
```

```
[1] "Fri Oct 21 06:37:04 2005"
```

The default order is day name, month name (both abbreviated), day of the month, hour (24-hour clock), minute, second (separated by colons) then the year. You can convert Sys.time to an object that inherits from class POSIXlt like this:

```
date<- as.POSIXlt(Sys.time())
```

You can use the element name operator $ to extract parts of the date and time from this object using the following names: sec, min, hour, mday, mon, year, wday, yday and isdst (with obvious meanings except for mday (=day number within the month), wday (day of the week starting at 0 = Sunday), yday (day of the year after 1 January = 0) and isdst which means 'is daylight savings time in operation?' with logical 1 for TRUE or 0 for FALSE). Here we extract the day of the week (date$wday = 0 meaning Sunday) and the Julian date (day of the year after 1 January as date$yday)

```
date$wday
```

```
[1] 0
```

```
date$yday
```

```
[1] 295
```

for 23 October. Use unclass with unlist to view all of the components of date:

```
unlist(unclass(date))
```

```
sec  min  hour  mday  mon  year  wday  yday  isdst
 42   17    10    23    9   105     0   295      1
```

Note that the month of October is 9 (not 10) because January is scored as month 0, and years are scored as post-1900.

Calculations with dates and times

You can do the following calculations with dates and times:

- time + number
- time – number
- time1 – time2
- time1 'logical operation' time2

where the logical operations are one of ==, !=, <, <=, '>' or >=. You can add or subtract a number of seconds or a difftime object (see below) from a date-time object, but you cannot add two date-time objects. Subtraction of two date-time objects is equivalent to using difftime (see below). Unless a time zone has been specified, POSIXlt objects are interpreted as being in the current time zone in calculations.

The thing you need to grasp is that you should convert your dates and times into POSIXlt objects *before* starting to do any calculations. Once they are POSIXlt objects, it is straightforward to calculate means, differences and so on. Here we want to calculate the number of days between two dates, 22 October 2003 and 22 October 2005:

```
y2<-as.POSIXlt("2003-10-22")
y1<-as.POSIXlt("2005-10-22")
```

Now you can do calculations with the two dates:

```
y1-y2
```

```
Time difference of 731 days
```

Note that you cannot *add* two dates. It is easy to calculate differences between times using this system. Note that the dates are separated by hyphens whereas the times are separated by colons:

```
y3<-as.POSIXlt("2005-10-22 09:30:59")
y4<-as.POSIXlt("2005-10-22 12:45:06")
y4-y3
```

```
Time difference of 3.235278 hours
```

The difftime function

Working out the time difference between to dates and times involves the difftime function, which takes two date-time objects as its arguments. The function returns an object of class difftime with an attribute indicating the units. How many days elapsed between 15 August 2003 and 21 October 2005?

```
difftime("2005-10-21","2003-8-15")
```

```
Time difference of 798 days
```

If you want only the number of days, for instance to use in calculation, then write

```
as.numeric(difftime("2005-10-21","2003-8-15"))
```

```
[1] 798
```

For differences in hours include the times (colon-separated) and write

```
difftime("2005-10-21 5:12:32","2005-10-21 6:14:21")
```

```
Time difference of -1.030278 hours
```

The result is negative because the first time (on the left) is before the second time (on the right). Alternatively, you can subtract one date-time object from another directly:

```
ISOdate(2005,10,21)-ISOdate(2003,8,15)
```

```
Time difference of 798 days
```

You can convert character stings into difftime objects using the as.difftime function:

```
as.difftime(c("0:3:20", "11:23:15"))
```

```
Time differences of 3.333333, 683.250000 mins
```

You can specify the format of your times. For instance, you may have no information on seconds, and your times are specified just as hours (format %H) and minutes (%M). This is what you do:

```
as.difftime(c("3:20", "23:15", "2:"), format= "%H:%M")
```

```
Time differences of 3.333333, 23.250000, NA hours
```

Because the last time in the sequence '2:' had no minutes it is marked as NA.

The strptime function

You can 'strip a date' out of a character string using the strptime function. There are functions to convert between character representations and objects of classes POSIXlt and POSIXct representing calendar dates and times. The details of the formats are system-specific, but the following are defined by the POSIX standard for strptime and are likely to be widely available. Any character in the format string other than the % symbol is interpreted literally.

%a Abbreviated weekday name

%A Full weekday name

%b Abbreviated month name

%B Full month name

%c Date and time, locale-specific

%d Day of the month as decimal number (01–31)

%H Hours as decimal number (00–23) on the 24-hour clock

%I Hours as decimal number (01–12) on the 12-hour clock

%j Day of year as decimal number (001–366)

%m Month as decimal number (01–12)

%M Minute as decimal number (00–59)

%p AM/PM indicator in the locale

%S Second as decimal number (00–61, allowing for two 'leap seconds')

%U Week of the year (00–53) using the first Sunday as day 1 of week 1

%w Weekday as decimal number (0–6, Sunday is 0)

%W Week of the year (00–53) using the first Monday as day 1 of week 1

%x Date, locale-specific

%X Time, locale-specific

%Y Year with century

%Z Time zone as a character string (output only)

Where leading zeros are shown they will be used on output but are optional on input.

Dates in Excel spreadsheets

The trick is to learn how to specify the format of your dates properly in strptime. If you had dates (and no times) in a dataframe in Excel format (day/month/year)

```
excel.dates <- c("27/02/2004", "27/02/2005", "14/01/2003",
                 "28/06/2005", "01/01/1999")
```

then the appropriate format would be "%d/%m/%Y" showing the format names (from the list above) and the 'slash' separators / (note the upper case for year %Y; this is the unambiguous year including the century, 2005 rather than the potentially ambiguous 05 for which the format is %y). To turn these into R dates, write

```
strptime(excel.dates,format="%d/%m/%Y")
```

```
[1] "2004-02-27" "2005-02-27" "2003-01-14" "2005-06-28" "1999-01-01"
```

Here is another example, but with years in two-digit form (%y), and the months as abbreviated names (%b) and no separators:

```
other.dates<- c("1jan99", "2jan05", "31mar04", "30jul05")
strptime(other.dates, "%d%b%y")
```

```
[1] "1999-01-01" "2005-01-02" "2004-03-31" "2005-07-30"
```

You will often want to create POSIXlt objects from components stored in vectors within dataframes. For instance, here is a dataframe with the hours, minutes and seconds from an experiment with two factor levels in separate columns:

```
times<-read.table("c:\\temp\\times.txt",header=T)
times
```

```
    hrs  min  sec  experiment
 1    2   23    6           A
 2    3   16   17           A
 3    3    2   56           A
 4    2   45    0           A
 5    3    4   42           A
 6    2   56   25           A
 7    3   12   28           A
 8    1   57   12           A
 9    2   22   22           B
10    1   42    7           B
11    2   31   17           B
12    3   15   16           B
13    2   28    4           B
14    1   55   34           B
15    2   17    7           B
16    1   48   48           B
```

```
attach(times)
```

Because the times are not in POSIXlt format, you need to paste together the hours, minutes and seconds into a character string with colons as the separator:

```
paste(hrs,min,sec,sep=":")
```

```
 [1]  "2:23:6"  "3:16:17"  "3:2:56"   "2:45:0"   "3:4:42"  "2:56:25"  "3:12:28"
 [8]  "1:57:12"  "2:22:22"  "1:42:7"  "2:31:17"  "3:15:16"   "2:28:4"  "1:55:34"
[15]  "2:17:7"  "1:48:48"
```

Now save this object as a difftime vector called duration:

```
duration<-as.difftime (paste(hrs,min,sec,sep=":"))
```

Then you can carry out calculations like mean and variance using the tapply function:

```
tapply(duration,experiment,mean)
```

```
       A         B
2.829375  2.292882
```

Calculating time differences between the rows of a dataframe

A common action with time data is to compute the time difference between successive rows of a dataframe. The vector called duration created above is of class difftime and contains 16 times measured in decimal hours:

```
class(duration)
```

```
[1] "difftime"
```

```
duration
```

```
Time differences of 2.385000, 3.271389, 3.048889, 2.750000, 3.078333,
2.940278, 3.207778, 1.953333, 2.372778, 1.701944, 2.521389, 3.254444,
2.467778, 1.926111, 2.285278, 1.813333 hours
```

We can compute the differences between successive rows using subscripts, like this

```
duration[1:15]-duration[2:16]
```

```
Time differences of -0.8863889, 0.2225000, 0.2988889, -0.3283333,
0.1380556, -0.2675000, 1.2544444, -0.4194444, 0.6708333, -0.8194444,
-0.7330556, 0.7866667, 0.5416667, -0.3591667, 0.4719444 hours
```

You might want to make the differences between successive rows into part of the dataframe (for instance, to relate change in time to one of the explanatory variables in the dataframe). Before doing this, you need to decide on the row in which to put the first of the differences. Is the change in time between rows 1 and 2 related to the explanatory variables in row 1 or row 2? Suppose it is row 1 that we want to contain the first time difference (-0.886). Because we are working with differences (see p. 719) the vector of differences is shorter by one than the vector from which it was calculated:

```
length(duration[1:15]-duration[2:16])
```

```
[1] 15
```

```
length(duration)
```

```
[1] 16
```

so we need to add one 'NA' to the bottom of the vector (in row 16).

```
diffs<-c(duration[1:15]-duration[2:16],NA)
diffs
```

```
 [1] -0.8863889  0.2225000  0.2988889 -0.3283333  0.1380556 -0.2675000
 [7]  1.2544444 -0.4194444  0.6708333 -0.8194444 -0.7330556  0.7866667
[13]  0.5416667 -0.3591667  0.4719444          NA
```

Now we can make this new vector part of the dataframe called times:

```
times$diffs<-diffs
times
```

```
    hrs  min  sec  experiment       diffs
1     2   23    6           A  -0.8863889
2     3   16   17           A   0.2225000
3     3    2   56           A   0.2988889
4     2   45    0           A  -0.3283333
5     3    4   42           A   0.1380556
6     2   56   25           A  -0.2675000
7     3   12   28           A   1.2544444
8     1   57   12           A  -0.4194444
9     2   22   22           B   0.6708333
10    1   42    7           B  -0.8194444
11    2   31   17           B  -0.7330556
12    3   15   16           B   0.7866667
13    2   28    4           B   0.5416667
14    1   55   34           B  -0.3591667
15    2   17    7           B   0.4719444
16    1   48   48           B          NA
```

There is more about dates and times in dataframes on p. 126.