Dates in Dataframes

There is an introduction to the complexities of using dates and times in dataframes on pp. 89–95. Here we work with a simple example:

```
nums<-read.table("c:\\temp\\sortdata.txt",header=T)
attach(nums)
names(nums)
```

```
[1] "name" "date" "response" "treatment"
```

The idea is to order the rows by date. The ordering is to be applied to all four columns of the dataframe. Note that ordering on the basis of our variable called date does not work in the way we want it to:

```
nums[order(date),]
```

```
        name        date     response  treatment
53    rachel  01/08/2003  32.98792196          B
65    albert  02/06/2003  38.41979568          A
6        ann  02/07/2003   2.86983693          B
10    cecily  02/11/2003   6.81467571          A
4        ian  02/12/2003   2.09505949          A
29   michael  03/05/2003  15.59890900          B
...
```

This is because of the format used for depicting date in the dataframe called nums: date is a character string in which the first characters are the day, then the month, then the year. When we sort by date, we typically want 2001 to come before 2006, May 2006 before September 2006 and 12 May 2006 before 14 May 2006. In order to sort by date we need first to convert our variable into date-time format using the strptime function (see p. 92 for details):

```
dates<-strptime(date,format="%d/%m/%Y")
dates
```

```
[1] "2003-08-25" "2003-05-21" "2003-10-12" "2003-12-02" "2003-10-18"
[6] "2003-07-02" "2003-09-27" "2003-06-05" "2003-06-11" "2003-11-02"
```

Note how strptime has produced a date object with year first, then a hyphen, then month, then a hyphen, then day which will sort into the desired sequence. We bind the new variable to the dataframe called nums like this:

```
nums<-cbind(nums,dates)
```

Now that the new variable is in the correct format the dates can be sorted as characters:

```
nums[order(as.character(dates)),1:4]
```

```
       name        date     response  treatment
49   albert  21/04/2003  30.66632632          A
63    james  24/04/2003  37.04140266          A
24     john  27/04/2003  12.70257306          A
33  william  30/04/2003  18.05707279          B
29  michael  03/05/2003  15.59890900          B
71      ian  06/05/2003  39.97237868          A
50   rachel  09/05/2003  30.81807436          B
```

Note the use of subscripts to omit the new dates variable by selecting only columns 1 to 4 of the dataframe. Another way to extract elements of a dataframe is to use the subset function with select like this:

```
subset(nums,select=c("name","dates"))
```

```
         name       dates
1      albert  2003-08-25
2         ann  2003-05-21
3        john  2003-10-12
4         ian  2003-12-02
5     michael  2003-10-18
...
...
73   georgina  2003-05-24
74   georgina  2003-08-16
75    heather  2003-11-14
76  elizabeth  2003-06-23
```

Selecting Variables on the Basis of their Attributes

In this example, we want to extract all of the columns from nums (above) that are numeric. Use sapply to obtain a vector of logical values:

```
sapply(nums,is.numeric)
```

```
 name   date  response  treatment  dates
FALSE  FALSE      TRUE      FALSE   TRUE
```

Now use this object to form the column subscripts to extract the two numeric variables:

```
nums[,sapply(nums,is.numeric)]
```

```
    response       dates
1 0.05963704  2003-08-25
2 1.46555993  2003-05-21
3 1.59406539  2003-10-12
4 2.09505949  2003-12-02
```

Note that dates is numeric but date was not (it is a factor, having been converted from a character string by the read.table function).

Using the match Function in Dataframes

The worms dataframe (above) contains fields of five different vegetation types:

```
unique(worms$Vegetation)
```

```
[1] Grassland  Arable  Meadow  Scrub  Orchard
```

and we want to know the appropriate herbicides to use in each of the 20 fields. The herbicides are in a separate dataframe that contains the recommended herbicides for a much larger set of plant community types:

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