Melbourne Holidays

RdR

2015-02-01

Calendar calculations are probably one of the oldest applications of math to everyday problems. In the past, real-world problems may have driven the calculation of calendars, but these days the reverse may also be true. Recently, I was looking at a time-series that seemed to have calendar effects: volumes of activities varied relative to public holidays.

Since I use R for my analysis, I needed a function to calculate the public holidays for Melbourne, which I'll describe here. Firstly, I use a the *lubridate* package for various date formats.

```
    # Melbourne public holidays
    require(lubridate, quietly=TRUE)
```

Next, we need a function to calculate the Easter date. Some years ago, there was a competition for the shortest formulas for Easter calculations in Excel. I translated Roger Friederich's contribution to R:

```
    # Easter calculation
    Easter <- function(year=1971){
        d <- ((year %% 19) * 18.37 -6) %% 29
        s <- as.numeric(as.Date(paste(year,3,d,sep="-"))) + 5
        s <- (s %/% 7 * 7 + 24)
        return(format(as.Date(s,origin="1970-01-01"),"%d-%m-%Y"))
    }
```

This function tells us that, for example, the date of Easter Sunday in 2015 falls on 2015-04-05.

A number of public holidays are calculated as some n^{th} weekday of a month. For example, Labour Day is the second Monday in February. So, we need an n^{th} day function:

```
    # calculate date of nth weekday in a month
    # example: 2nd Monday in the month
    # daynum is Sun=1, Mon=2, Tue=3,...
    nthday <- function(daynum, n, month, year){
        Monthstart <- dmy(paste(1,month,year,sep="-"))
        firstday <- wday(Monthstart)
        delta <- (daynum + 7 - firstday) %% 7 + (n-1)*7
        return(Monthstart + days(delta))
    }
```

Now we're ready to put it all together. The final function will produce a simple list of the "effective"" date of the holidays. Effective date is the the actual free workday that occurs (which may be the Monday or Tuesday after the public holiday if it falls on a weekend).

Unfortunately, there is an exception for ANZAC day, which does not result in a free workday if it happens to fall on a weekend. This ruling depends political rulings, so needs to be hard coded.

```
# get list of holidays for given year
    melbhols <- function(year=1971){
        hols <- dmy(paste( 1,1, year, sep="-"))
        hols <- c(hols, dmy(paste( 26,1, year, sep="-")))
        hols <- c(hols, nthday(2,2,3,year))

        EasterSunday <- dmy(Easter(year))
        hols <- c(hols, EasterSunday + ddays(-2))
        hols <- c(hols, EasterSunday + ddays(1))

        hols <- c(hols, dmy(paste(25,4,year,sep="-")))
        hols <- c(hols, nthday(2,2,6,year))
        hols <- c(hols, nthday(3,1,11,year))
        hols <- c(hols, dmy(paste(25,12,year,sep="-")))
        hols <- c(hols, dmy(paste(26,12,year,sep="-")))

        names(hols) <- c("NewYears","AusDay","LabourDay","GoodFriday",
                         "EasterMonday","Anzac","QueensBD","CupDay",
                         "Xmas","BoxingDay")

        # adjust for holidays on weekends
        nh <- length(hols)
        for(h in 1:nh){
            if(h != 6){  # days in lieu, except for ANZAC
                wd <- wday(hols[h])
                if(wd==7) hols[h] <- hols[h] + ddays(2)
                if(wd==1) hols[h] <- hols[h] + ddays(1)
            }
        }
        if(hols[nh]==hols[nh-1]) hols[nh] <- hols[nh] + ddays(1)

        return(format(as.Date(hols),"%d-%m-%Y"))
    }
```

So, for 2015 the public holidays are as follows:

```
    melbhols(2015)
```

```
##     NewYears       AusDay    LabourDay   GoodFriday EasterMonday
## "01-01-2015" "26-01-2015" "09-03-2015" "03-04-2015" "06-04-2015"
##        Anzac     QueensBD       CupDay         Xmas    BoxingDay
## "25-04-2015" "08-06-2015" "03-11-2015" "25-12-2015" "28-12-2015"
```