88.77 An extended approach to the Julian and the Gregorian calendar

Introduction

Pope Gregory XIII proclaimed the Gregorian calendar in 1582, correcting for the accumulated discrepancy between the Julian calendar and the equinox as of that date. In the Julian calendar, every fourth year is a leap year in which February has 29, rather than 28, days, but in the Gregorian, years divisible by 100 are not leap years unless they are also divisible by 400. England and its colonies started using the Gregorian calendar in 1752. Specifically, the British, and the inhabitants of their colonies, went to bed on Wednesday 2nd September 1752 and arose next morning to Thursday 14th September 1752. Sweden changed in 1753 and a number of (mostly Orthodox) countries did not change until the early 20th century (Russia in 1918).

There are two existing methods to find the day of the week, given the date.

Zeller's algorithm [see 1]

The days of the week are numbered from 0, for Saturday, to 6, for Friday. Months are numbered from 3, for March, to 14, for February. The year is assumed to begin in March; this means, for example, that January 1995 is to be treated as month 13 of 1994.

For the Julian calendar, use

\[ w = 5 + d + \left\lfloor \frac{26(m + 1)}{10} \right\rfloor + \left\lfloor \frac{5y}{4} \right\rfloor - c \mod 7 \]

and for the Gregorian calendar, use

\[ w = d + \left\lfloor \frac{26(m + 1)}{10} \right\rfloor + \left\lfloor \frac{5y}{4} \right\rfloor + \left\lfloor \frac{c}{4} \right\rfloor - 2c \mod 7 \]

where:
- \( w \) is the number of the day of the week
- \( d \) is the day of the month
- \( m \) is the number of the month
- \( y \) is the final two digits of the month
- \( c \) is the first two digits of the year - the century.

For example, to find the day of the week for 28 December 2002 (Gregorian), substitute \( d = 28 \), \( m = 12 \), \( c = 20 \) and \( y = 2 \) to obtain \( w = 0 \), a Saturday.

The key-value method

This method, which works for the Gregorian calendar only, uses codes for different months and years to speed up the calculation of the day of the week.
\[ w \equiv d + \left\lfloor \frac{5y}{4} \right\rfloor + M + C \pmod{7} \]

where:  
- \( M \) is the month code, found from Table 1; if the date is in January or February of a leap year, subtract 1 from this code. 
- \( C \) is the century code, found from Table 2.

<table>
<thead>
<tr>
<th>Jan</th>
<th>Feb</th>
<th>Mar</th>
<th>Apr</th>
<th>May</th>
<th>Jun</th>
<th>Jul</th>
<th>Aug</th>
<th>Sept</th>
<th>Oct</th>
<th>Nov</th>
<th>Dec</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4</td>
<td>4</td>
<td>0</td>
<td>2</td>
<td>5</td>
<td>0</td>
<td>3</td>
<td>6</td>
<td>1</td>
<td>4</td>
<td>6</td>
</tr>
</tbody>
</table>

**TABLE 1**

<table>
<thead>
<tr>
<th>Year</th>
<th>1700</th>
<th>1800</th>
<th>1900</th>
<th>2000</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code</td>
<td>4</td>
<td>2</td>
<td>0</td>
<td>6</td>
</tr>
</tbody>
</table>

**TABLE 2**

For example, to find the day for 28 December 2002 (Gregorian), substitute \( d = 28, y = 2, M = 6 \) and \( C = 6 \) to obtain, again, a Saturday.

The key-value method can be derived from our alternative approach below.

**An alternative approach**

These methods only find the day of the week. My aim is to develop alternative formulae which are easier to use and also allow one to find the date, month and year when the other information is given.

These formulae are to be of the following form:

Julian:  
\[ w \equiv d + f(m) + g(y) + h_f(c) + \delta_f \pmod{7} \]

Gregorian:  
\[ w \equiv d + f(m) + g(y) + h_G(c) + \delta_G \pmod{7} \]

where the functions \( f, g \) and \( h \) are to be determined, and where \( w, d, m, y \) and \( c \) stand, in some way to be made specific, for weekday, date, month, year and century. It is useful also to define the fullyear number \( Y = 100c + y \). The \( \delta_f, \delta_G \) are constants that align the weekday sequence correctly.

**Dealing with the year**

We begin with the function \( g(y) \). This function must model the way in which \( w \) increases with \( y \). For January 1st after a normal year, \( w \) should increase by 1 as \( Y \) increases by 1, but after a leap year, it should increase by 2. Thus, in the Julian calendar, \( \left\lfloor \frac{5y}{4} \right\rfloor \) differs by a constant (modulo 7) from the weekday number of 1st January year \( Y + 1 \). But

\[ \left\lfloor \frac{5Y}{4} \right\rfloor = \left\lfloor \frac{500c + 5y}{4} \right\rfloor = 125c + \left\lfloor \frac{5y}{4} \right\rfloor \equiv -c + \left\lfloor \frac{5y}{4} \right\rfloor \pmod{7}, \]

so \( g(y) = \left\lfloor \frac{5y}{4} \right\rfloor \) and \( h_f(c) = -c \).

However, there are two caveats. First, we must ensure that days before February 29th in a leap year are treated differently from days after that date.
This will be discussed under the derivation of \( f(m) \). Secondly, there is the problem of centenary years in the Gregorian calendar. For the Julian calendar, it would be adequate simply to define \( y \) as the four-digit year, but for the Gregorian, this will not do. By the same argument as found that \( \frac{[2y]}{4} \) differs by a constant (modulo 7) from the weekday number of 1st January year \( Y + 1 \) in the Julian calendar, we find that

\[
\left[ \frac{5Y}{4} \right] - \left[ \frac{Y}{100} \right] + \left[ \frac{Y}{400} \right] = 125c + \left[ \frac{5y}{4} \right] - c + \left[ \frac{c}{4} \right] \equiv \left[ \frac{5Y}{4} \right] - 2c + \left[ \frac{c}{4} \right] \pmod{7}
\]

differs by a constant (modulo 7) from the weekday number of 1st January year \( Y + 1 \) in the Gregorian calendar, so \( g(y) = \left[ \frac{5y}{4} \right] \), and we may choose \( h_G(c) = -2c + \left[ \frac{c}{4} \right] \), for the Gregorian calendar, where, as usual, we are going to evaluate it modulo 7. When we add this function to \( g(y) \) we obtain the correct formula for dealing with the full year.

**Dealing with the month**

Finally we must construct the month function \( f(m) \). We wish to avoid the rather artificial use of month numbers 3 to 14 in Zeller's algorithm so we number the months from January as 1 to December as 12. The function will count the shift in the weekday appropriate to each month of the year. Clearly, in a normal year, \( f(1) = 0 \), and, for each successive month, we must allow for the fact that the preceding month advances the weekday number according to the following table:

<table>
<thead>
<tr>
<th>Month</th>
<th>Jan</th>
<th>Feb</th>
<th>Mar</th>
<th>Apr</th>
<th>May</th>
<th>Jun</th>
<th>Jul</th>
<th>Aug</th>
<th>Sept</th>
<th>Oct</th>
<th>Nov</th>
<th>Dec</th>
</tr>
</thead>
<tbody>
<tr>
<td>( f(m) )</td>
<td>0</td>
<td>3</td>
<td>3</td>
<td>6</td>
<td>1</td>
<td>4</td>
<td>6</td>
<td>2</td>
<td>5</td>
<td>0</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>For leap years</td>
<td>6</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**TABLE 3**

The final row of the table is to allow for the fact that, in a leap year, the additional \( w \) increment for 1st January year \( Y + 1 \), compared with 1st January year \( Y \), applies to days in year \( Y \), except for those in January and February. For the moment we ignore this and construct a function which produces the values in the row labelled \( f(m) \). One such function is given by:

\[
f(m) = \begin{cases} 
\left[ \frac{5m-4}{2} \right] & 1 \leq m < 2 \\
\left[ \frac{5m+6}{2} \right] & 3 \leq m < 8 \\
\left[ \frac{5m+7}{2} \right] & 9 \leq m < 12.
\end{cases}
\]

It is possible to express this in a somewhat convoluted closed form:

\[
f(m) = \left[ 3 + 2 \left\{ \frac{1}{2} + \frac{1}{m} \right\} + \frac{5m + \left[ \frac{m}{2} \right]}{2} \right].
\]

It is easy to check by calculation that this does produce the correct figures.
Finally, to take account of the leap years, one introduces a new variable $L$ which takes the value 1 in a leap year and 0 otherwise, and defines

$$f(m) = \left[ 3 + 2 \left\lfloor 1/2 + \frac{1}{m} \right\rfloor + \frac{5m + \left\lfloor \frac{11}{2} \right\rfloor}{2} \right] - L \left\lfloor 1/2 + \frac{1}{m} \right\rfloor$$

which reduces the value of $f$ by 1 in the first two months of a leap year.

The formula

One can now put the ingredients together to produce the final function. Our arguments have given us formulae that differ (modulo 7) by a constant from the desired values. There is a Julian constant $\delta_J$, and a Gregorian constant $\delta_G$. To discover them, we use the data for the British changeover, given in the introduction. It is computationally convenient to use $\left\lfloor \frac{y}{4} \right\rfloor = y + \frac{y}{4}$. Then, for the Julian calendar, $4 \equiv 2 + 5 + 52 + 13 - 17 + \delta_J \equiv 3 - 1 + 4 + \delta_J$, so $\delta_J = -2$. For the Gregorian calendar, $5 \equiv 14 + 5 + 52 + 13 - 2 \times 17 + \left\lfloor \frac{12}{4} \right\rfloor + \delta_G \equiv 5 + 3 - 1 + 1 + 4 + \delta_G$, so $\delta_G = 0$.

Thus:

Julian:

$$w \equiv d + f(m) - (m < 3)(4 \mid y) + y + \left\lfloor \frac{y}{4} \right\rfloor - c - 2 \equiv d + \left[ 3 + 2 \left\lfloor 1/2 + \frac{1}{m} \right\rfloor + \frac{5m + \left\lfloor \frac{11}{2} \right\rfloor}{2} \right] - L \left\lfloor 1/2 + \frac{1}{m} \right\rfloor + \left\lfloor \frac{5y}{4} \right\rfloor - c - 2 \pmod{7}$$

Gregorian:

$$w \equiv d + f(m) - (m < 3)\{(y \neq 0)(4 \mid y) + (y = 0)(4 \mid c)\} + y + \left\lfloor \frac{y}{4} \right\rfloor - 2c + \left\lfloor \frac{c}{4} \right\rfloor \equiv d + \left[ 3 + 2 \left\lfloor 1/2 + \frac{1}{m} \right\rfloor + \frac{5m + \left\lfloor \frac{11}{2} \right\rfloor}{2} \right] - L \left\lfloor 1/2 + \frac{1}{m} \right\rfloor + \left\lfloor \frac{5y}{4} \right\rfloor + 5\left(c - 4 \left\lfloor \frac{c}{4} \right\rfloor \right) \pmod{7}$$

again (mod 7). In both of these formulae, $w$ and $d$ are the usual weekday and date functions. The month $m$ is defined normally as between 1 and 12. The full year $Y$ is split between $y$ and $c$ as already mentioned.

Using the formula

It is straightforward to find the weekday given the other data and the date using the other data. Given everything except the month, the best that can be done is to find and refer to Table 3. There are many years within the same century which share the same weekday, date and month. Interested readers are invited to refer to my website

http://www.geocities.com/sohaelbabwani/algorithm.html

for further details.

Acknowledgement

The author is grateful to the proof-reader and editor for their help in structuring this note.
88.78 Optimal scheduling to minimise waiting time

An interesting use of mathematics is when it can be applied to an everyday situation to help make a better choice from a number of available courses of action. One such example is considered by [1], in which various gambling opportunities are evaluated by considering their house margins.

In this paper, the scenario is one of a waiting room, e.g. at a hospital or medical facility, where there are a number of patients waiting to be seen. It is assumed that none of their ailments are life-threatening (so there are no priorities), and that all patients must be attended to on the day.

In particular, suppose that there are \( n \) patients who are scheduled to see a single medical practitioner, and they have been previously assessed by nursing staff to have estimated treatment times of \( x_1, x_2, x_3, \ldots, x_n \) where, without loss of generality, \( x_1 \leq x_2 \leq x_3 \leq \ldots \leq x_n \). That is, patient \( i \) is defined as the one with the \( i \)th shortest waiting time. Since each patient must be attended to, the practitioner does not care in which order the patients are seen since the total time they will spend on them is always a constant \( \sum_{i=1}^{n} x_i < d \), the treatment hours of the day.

From the patients' point of view, the aim is therefore to minimise the mean time that the patients must wait. The actual waiting time will naturally vary from patient to patient, since the first patient seen will always have zero waiting time while the last patient seen will have to wait until all \( n - 1 \) of the others have finished.

Before considering the general case, it is instructive to consider the scenarios for only a few patients.

One patient \((n = 1)\)

This is the simplest case where there is no waiting time, since this patient is seen first since they are the only one.

Two patients \((n = 2)\)

There are only two possible orderings of two patients with the resulting waiting times as shown in Table 1.

<table>
<thead>
<tr>
<th>Order of patients</th>
<th>Total waiting time</th>
<th>Mean waiting time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1, 2</td>
<td>( x_1 )</td>
<td>( \frac{1}{2}x_1 )</td>
</tr>
<tr>
<td>2, 1</td>
<td>( x_2 )</td>
<td>( \frac{1}{2}x_2 )</td>
</tr>
</tbody>
</table>

**TABLE 1**: Total and mean waiting times for two patients