Dates and Times in Matlab and Java

Fall 2016, UC Berkeley, ME C231A and EECS C220B

Contents

How does Matlab represent dates and times?

The command `datenum` converts a date vector into a single number, called the **serial date**. As a reference, the serial date 1 corresponds to the date Jan-1-0000. As described in the `datenum` documentation, *the year 0000 is merely a reference point and is not intended to be interpreted as a real year*.

```
format compact
```

datenum(Year, month, day)

This returns an integer (class in `double`), again with Jan-1-0000 returning 1.

```
datenum(0,1,1)
```

```
ans =
     1
```

```
class(datenum(0,1,1))
```

```
ans =
double
```

Advancing to Feb-1-0000 should give 31+1 = 32

```
datenum(0,2,1)
```

```
ans =
    32
```

Advancing to March-1-0000 should give 31 + 29 + 1 = 61, since February in the year 0000 would be a 29-day month.

```
datenum(0,3,1)
```

```
ans =
    61
```

Advancing to March-1-0001 should give 61 + 365 = 426.

```
datenum(1,3,1)
```

```
ans =
   426
```

By this counting scheme, September 6, 2016 is day 736,579.

```
datenum(2016, 9, 6)
```

```
ans =
      736579
```

datenum(Year, month, day, hour, minute, second)

For nonzero `hour`, or `minute` or `second`, this will return a non-integer value, representing the fraction of the day past midnight. So, at noon on Jan-1-0000, it returns 1.5

```
datenum(0,1,1,12,0,0)
```

```
ans =
   1.5000e+00
```

6 hours later, at 6:00PM, give 1.75

```
datenum(0,1,1,18,0,0)
```

```
ans =
   1.7500e+00
```

`datenum` also accepts a 1-by-6 array (instead of 6 arguments). The command `clock` returns the current time (as set by your computer's clock) in the [Year Month Day Hour Minute Second] format.

```
clock
```

```
ans =
   2.0160e+03   9.0000e+00   7.0000e+00   2.0000e+00   5.3000e+01   3.7890e+00
```

Convert current time to the date-number,

```
datenum(clock)
```

```
ans =
   7.3658e+05
```

Note that `datenum` and `clock` are not referenced to any standard time zone. These are just a reflection of what the computer's clock says, relative to Jan 1, 0000. If two computers are the same room, one set to Eastern time (eg., NYC) and one set to Pacific time (ie. so the computers' clocks are 3 hours apart) both run `datenum(clock)` at the same instant, their answers will differ by 0.125 (1/8th of a day).

How does JAVA represent dates and times?

JAVA uses millseconds since 00:00:00 UTC on January 1, 1970. Note it is referenced to UTC. UTC is 8 hours ahead of Pacific time (7 hours during daylight savings time). The command `java.lang.System.currentTimeMillis()` returns the number of milliseconds since 00:00:00 UTC on January 1, 1970. I am not sure how Java knows what time zone I am in, to interpret my computer's clock correctly, but somewhere that is available from the operating system.

```
cJ = java.lang.System.currentTimeMillis()
```

```
cJ =
   1.4732e+12
```

It also returns a `double`

```
class(cJ)
```

```
ans =
double
```

Converting from one (Matlab, PDT) to another (Java, UTC)

Suppose we take the current time with Matlab (`clock`) and convert it to a serial date (with `datenum`). In order to compare to the Java return value, we only want to account for time since the beginning of the day on January 1, 1970, so we should subtract off `datenum(1970,1,1)`. Then since UTC is 7 hours ahead of PDT, we should add 7/24. Finally, convert this to milliseconds, multiplying by 1000*60*60*24 (the number of millseconds in a day). This number should agree (quite closely - at most a few milliseconds apart, in the 1-trillion millseconds since January 1970) with the result from `java.lang.System.currentTimeMillis()`.

```
cM = ((datenum(clock)-datenum(1970,1,1))+7/24)*1000*60*60*24;
cJ = java.lang.System.currentTimeMillis();
[cM cJ cJ-cM]
```

```
ans =
   1.4732e+12   1.4732e+12  -3.4180e-03
```

Note, this is not to say the clock on the computer is accurate in any sense. It merely says that the manner in which Java and Matlab are accessing it, and returning some documented representation of it are in agreement. This conversion (and its inverse) are implemented in the `PDTtoUTC` and `UTCtoPDT`

```
type PDTtoUTC
```

```
function UTCjavaRep = PDTtoUTC(PDTmatlabRep)
% Convert Matlab serial date in PDT to UTC time in Java (milliseconds since
% 00:00:00 UTC Jan 1, 1970).
UTCjavaRep = (PDTmatlabRep - datenum(1970,1,1) + 7/24)*(86400*1000);
% Attribution: Sarah Koehler, skoehler@berkeley.edu
```

```
type UTCtoPDT
```

```
function PDTmatlabRep = UTCtoPDT(UTCjavaRep)
% Convert UTC time in Java (milliseconds since 00:00:00 UTC Jan 1, 1970) to
% Matlab serial date in PDT.
PDTmatlabRep = UTCjavaRep/(86400*1000) - 7/24 + datenum(1970,1,1);
% Attribution: Sarah Koehler, skoehler@berkeley.edu
```

Obviously, there is simple code for Pacific Standard time (PST) to UTC conversions.

```
type PSTtoUTC
```

```
function UTCjavaRep = PSTtoUTC(PSTmatlabRep)
% Convert Matlab serial date in PST to UTC time in Java (milliseconds since
% 00:00:00 UTC Jan 1, 1970).
UTCjavaRep = (PSTmatlabRep - datenum(1970,1,1) + 8/24)*(86400*1000);
% Attribution: Sarah Koehler, skoehler@berkeley.edu
```

```
type UTCtoPST
```

```
function PSTmatlabRep = UTCtoPST(UTCjavaRep)
```

```
% Convert UTC time in Java (milliseconds since 00:00:00 UTC Jan 1, 1970) to
% Matlab serial date in PST.
PSTmatlabRep = UTCjavaRep/(86400*1000) - 8/24 + datenum(1970,1,1);
% Attribution: Sarah Koehler, skoehler@berkeley.edu
```

Attribution

UC Berkeley, Fall 2016. Used in ME C231A and EECS C220B for reference. Relevant when extracting building data from sMAP.

Published with MATLAB® R2016a