Time Standards and R

dmm

December 1, 2015

Dates and times can often be difficult for the software engineer. One source of difficulty can be the plethora of time zones and daylight savings times, and the subjectivity of those human-contrivances to the commercial and political whim of regions throughout the world, year after year. Another source is the number of different time standards. Of those two sources, the former might be perceived as the more exciting. This paper is about the latter, and the potential impact on the R user.

Time Standards

A cursory internet search for "time standard" might yield the newspaper in Eureka, California. (Nice town.) But back to the topic at hand, the best site by far these days is that maintainted by the folks at the Time Services Deparment of the United States Navy ("Tycho") (http://tycho.usno.navy.mil/).[1]

Tycho lists these Top Six time standards:

1. UT1
2. TAI
3. UTC
4. TDT
5. TCG
6. Siderial

We will cover the first three in this paper.

Universal Time

Universal Time (UT) is a family of time scales based on the rotation of the earth. The unit of Universal Time is the `mean solar day`. The `mean solar second` is defined to be one 86400th (86400=60x60x24) of the `mean solar day`.[2] Due to the gravitational pull of the moon and the motions of the tides, the earth is taking longer to orbit the sun.

"Greenwich Mean Time" was also based on the rotation of the earth. Over time it became clear that GMT as measured at the Royal Observatory was subject to an unsatisfactory amount of variability due to many factors, including polar shifts, tectonic plate movements, and the slowing of the earth's orbit around the sun due to the moon's gravitational pull and its impact on the tides. Eventually the international community settled on a more stable time scale (see below).

Nevertheless, people do find it useful to look upon a "day" in terms of the rotation of the earth. Therefore Universal Time is still in use today. Its current version is called UT1 and is the result of a mathematical formula based on observations of the sun and other astronomical objects.[3]

[1] My Greek friends are mixed regarding the translation of $\tau\upsilon\chi\omega\nu$ as "hitting the target."

[2] http://tycho.usno.navy.mil/leapsec.html

[3] For more information, see "The New Definition of Universal Time" by Aoki et. al., *Astronomy and Astrophysics*, vol. 105, no. 2, Jan. 1982, p. 359-361.

Atomic Time

The search for an intrinsically stable time scale led to in investigation of oscillation properties of various atoms, eventually settling on cesium 133. The `SI second` (Systeme International) is time it takes the cesium 133 atom to oscillate through 9,192,631,770 cycles under strict measurement conditions.[4] The Bureau International des Poids et Mesures (BIPM) averages readings from a large number of cesium 133 clocks in over fifty laboratories around the world, and publishes the definitive SI second for worldwide consumption.[5]

This time standard is known as `TAI`, *International Atomic Time.*

aka "Unix time" (also known as POSIX time or Epoch time)
https://en.wikipedia.org/wiki/Unix_time

Coordinated Universal Time (UTC)

"UTC is defined by the *CCIR Recommendation 460-4 (1986)*"[6]

The last time a mean solar day was precisely 86400 seconds (as atomically determined; see below) was in 1820.[7]

As straightforward a time system as UT1 is, it is not conducive to As satisfactory as that solution is for humans to keep track of, something much more reg

(or)

One was chosen and given the name "Universal Time."

A non-earth reference needed to be chosen.

, so that those zeniths could be recognized.

"Mean solar time") during the pointy months. But that is not rocket science. That After all, "Greenwich Mean Time" has always been the mean time it takes the sun to zenith each day (solar time) of=ver the course of a year. Unfortnately, that is no longer good eanough in terms of accuracy. That was recognized a long time ago and an international committee was formed.

The first resulting standard – in this paper and in Tycho's list – was that based on the definition of a "second."

[]

, which they refer with the more appropriate "Systems of Time":[8]

1. Atomic time
2. Universal time
3. Coordinated Universal Timeˆ["Wait. Not `CUT`?"

I would so much enjoyed to have been a fly on that wall.]

"Who cares," one hears, "my plot of" If in the past, problems attacked by computer solutions via statistical or analytical models tended to be dimensionally local so, Who cares?, that may be changed these days. In the past It is arguable that most analysis need not be concerned about time standards, time zones, or daylight

[4]See for instance http://tycho.usno.navy.mil.leapsec.html.
[5]See http://www.bipm.org/en/biipm/tai/tai.html
[6]tycho/leapsec.html
[7]http://tycho.usno.navy.mil.leapsec.html.
[8]http://tycho.usno.navy.mil/systime.html

time, so, who cares? However, with the ever interconnectedness of our world these days, more and more people see the importance of getting it straight.

If you ask people what the current time standard is, many people might say "Greenwich Mean Time" (GMT). But GMT has not been a *time standard* since the 1960's. Today, GMT is simply a *time zone* adopted by many countries along the prime meridian, 0° longitude. Time zones are typically an integral number of hours apart (1 hour, 3 hours, -5 hours), but fractional hour differences do exist, as with some of Australia's time zones.[9]

A *time standard* is an agreed-upon procedure for determining the time corresponding to an instant. Also known as *time systems,* the United States Naval Observatory lists seven different systems of time on its "tycho" website.[10] We will look at the first three of those systems.

Bibliography

A good site on the difference between TAI and UTC, definition of TAI-10, an algorithm based on TAI-10, and an argument against that.

http://www.madore.org/~david/computers/unix-leap-seconds.html

Footnote 2 says
A typical PC quartz seems to have an accuracy of about ten to fifty parts per million, i.e., a couple of seconds per day. Counting SI seconds without any attempt at correction gives UT1 an accuracy of about 2ms per day, i.e., 20 parts per billion, so that's around a thousand times better. Hence, without external assistance (say by NTP, an atomic clock or a GPS receiver), the typical PC quartz cannot see or hope to see the difference between universal time and atomic time.

these sites explain how months came about:
http://science.howstuffworks.com/science-vs-myth/everyday-myths/time6.htm

https://www.quora.com/Why-the-do-calender-months-have-such-varied-days-such-as-30-31-28-and-29

DUT1 https://en.wikipedia.org/wiki/DUT1 UTC is maintained via leap seconds, such that DUT1 remains within the range -0.9s < DUT1 < 0.9s. The reason for this correction is partly that the rate of rotation of the Earth is not constant, due to tidal braking and the redistribution of mass within the Earth, including its oceans and atmosphere, and partly because the SI second (as now used for UTC) was already, when adopted, a little shorter than the current value of the second of mean solar time.

Appendix

```
format(as.POSIXlt(tail(.leap.seconds,1), tz="UTC"), format="%Y-%m-%d %H:%M:%S")
```

```
## [1] "2015-07-01 00:00:00"
```

```
as.numeric(as.POSIXlt(tail(.leap.seconds,1), tz="UTC")) #1435708800
```

```
## [1] 1435708800
```

[9] For an entertaining video on strange time zones, see https://www.youtube.com/watch?v=uW6QqcmCfm8
[10] http://tycho.usno.navy/mil/systime.html. The word "tycho" comes from the greek for "hitting the mark." It is also the name of a Danish astronomer, Tycho Brahe (1546-1601).

```
a <- as.numeric(as.POSIXlt(tail(.leap.seconds,1), tz="UTC")) - 86400
format(as.POSIXlt(a, origin="1970-01-01", tz="UTC"), format="%Y-%m-%d %H:%M:%S")
```

```
## [1] "2015-06-30 00:00:00"
```

```
format(as.POSIXlt(a, origin="1970-01-01", tz="UTC"), format="%Y-%m-%d %H:%M:%S")
```

```
## [1] "2015-06-30 00:00:00"
```

```
d <- as.Date("2015-06-29")
d + 0:3
```

```
## [1] "2015-06-29" "2015-06-30" "2015-07-01" "2015-07-02"
```

```
t <- as.POSIXlt(d + 0:3)
format(t, format="%Y-%m-%d %H:%M:%S")
```

```
## [1] "2015-06-29 00:00:00" "2015-06-30 00:00:00" "2015-07-01 00:00:00"
## [4] "2015-07-02 00:00:00"
```

```
a <- as.numeric(t)
a
```

```
## [1] 1435536000 1435622400 1435708800 1435795200
```

```
a[1] + 86400*(0:3) # no leap second
```

```
## [1] 1435536000 1435622400 1435708800 1435795200
```