Dates and Times

James Balamuta

Departments of Informatics and Statistics
University of Illinois at Urbana-Champaign

December 10, 2018

On the Agenda

Date and Time Formats

"The only reason for time is so that everything doesn't happen at once."
— Albert Einstein

- *R* has the ability to interface with time information.
- The interface, as we will see, may not be the best but it is highly versatile.
- This is important in a world that is going more and more global.

Date and Time Formats

```
Sys.Date()                # Returns a date as R's Date object
```

```
## [1] "2018-12-10"
```

```
Sys.time()                # Returns both date & time at current locale as POSIXct
```

```
## [1] "2018-12-10 12:44:00 CST"
```

```
as.numeric(Sys.time()) # Seconds from UNIX Epoch (1970-01-01 00:00:00 UTC)
```

```
## [1] 1544467441
```

```
Sys.timezone()            # Time zone at current location
```

```
## [1] "America/Chicago"
```

Date and Time Formats - Failure of `characters`

- Frequently, dates and times will be given as `characters` within a `data.frame`
- Having dates as `characters` impedes ones ability to be able to use the time information in an analysis
  - For example: How long did it take for the help desk call to be completed?

```
# Bad Time Differencing:

time1 = "2017-07-10 10:25:44 CDT" # UNIX Time Stamp
time2 = "2017-07-10 15:25:44 CDT" # UNIX Time Stamp
time2 - time1
# Error in time2 - time1 : non-numeric argument to
# binary operator
```

Date and Time Formats - Time Operations

- Performing time operations requires that both dates are given as `POSIXct` object in *R*.

```
time1 = as.POSIXct("2017-07-10 10:25:44")
time2 = as.POSIXct("2017-07-10 15:25:44") # +5 Hours
time2 - time1
```

```
## Time difference of 5 hours
```

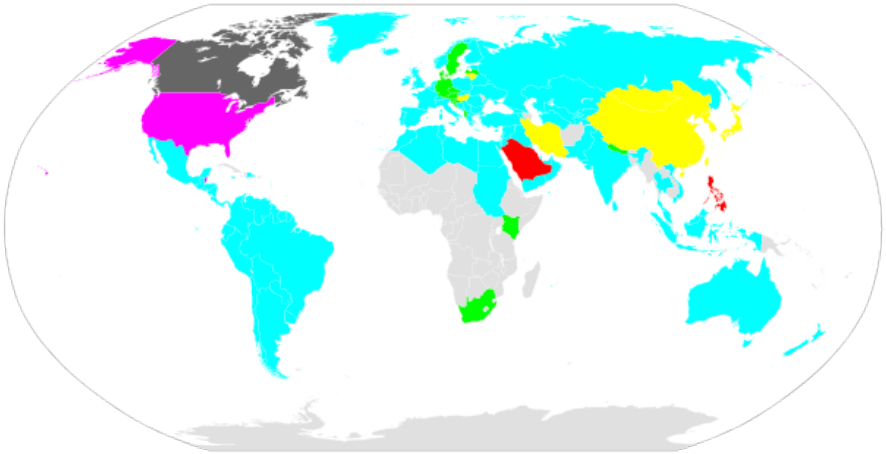
Note: Default format for `POSIXct` is `%Y-%m-%d %H:%M:%S`

Your Turn

How many days are in a semester if we have the dates of:

- First day: August 27th
- Last day: December 12th

The Date Format Around the World



Color	Date Format	Main Region	Population (Millions)
Cyan	DD/MM/YYYY	Australia, Russia	3295
Yellow	YYYY/MM/DD	China, Korea, Iran	1660
Magenta	MM/DD/YYYY	United States	320

Source: Date Formats by Country

Formats for Working with Dates

Format	Description	Example
%a	Abbreviated weekday name in the current locale	Mon
%A	Full weekday name in the current locale	Monday
%b	Abbreviated month name in the current locale	Dec
%B	Full month name in the current locale	December
%m	Month number (01-12)	12
%d	Day of the month as decimal number (01-31)	10
%e	Day of the month as decimal number (1–31)	10
%y	Year without century (00-99)	18
%Y	Year including century	2018

For more, see `?strptime`

Formating Non-Standard Dates

```
(yyyy_mm_dd = as.POSIXct("2017-07-10",
                         format = "%Y-%m-%e"))
```

```
## [1] "2017-07-10 CDT"
```

```
(dd_mm_yy = as.POSIXct("10/07/17",
                       format = "%e/%m/%y"))
```

```
## [1] "2017-07-10 CDT"
```

```
(mon_dd_yyyy = as.POSIXct("Jul 07, 2017",
                          format = "%b %e, %Y"))
```

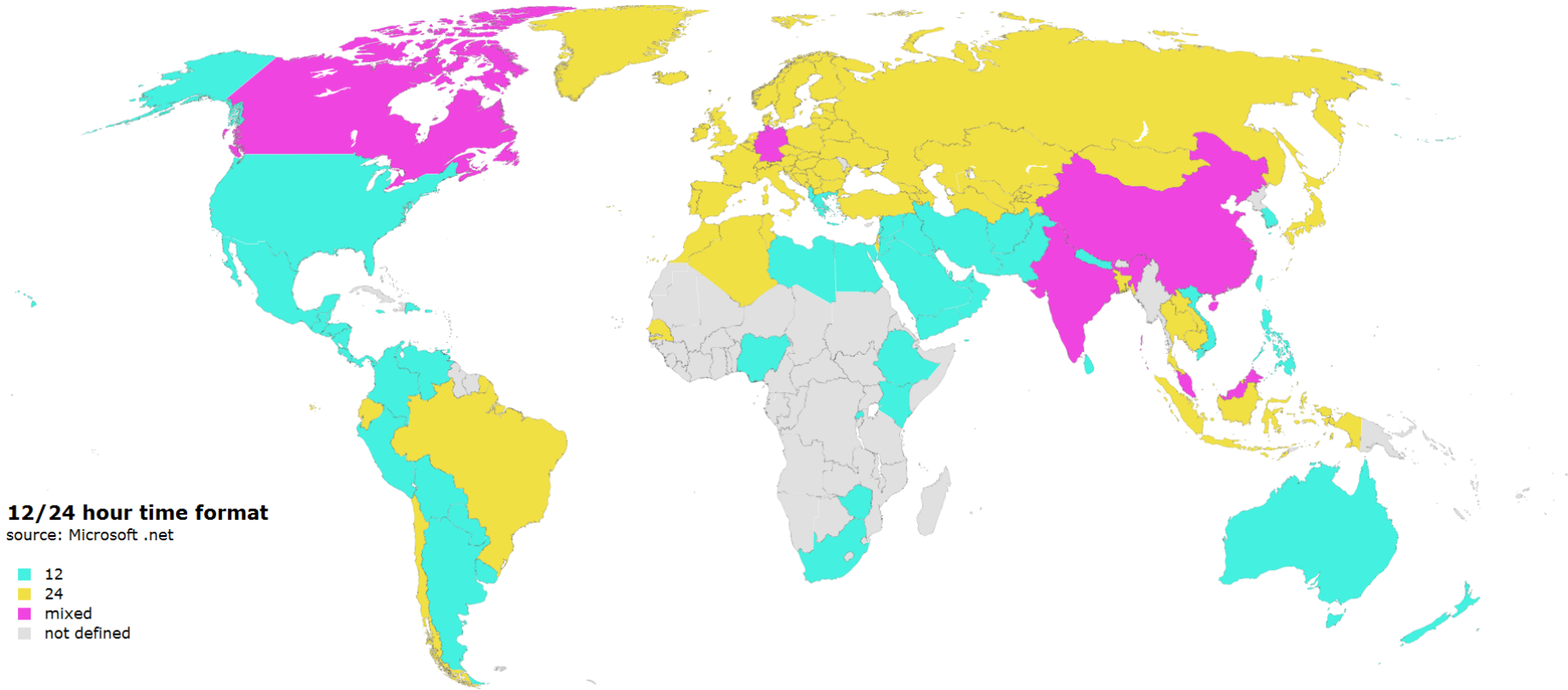
```
## [1] "2017-07-07 CDT"
```

Your Turn

How would you convert. . .

- 2017-07-10, e.g. YYYY/MM/DD
- September 10, 2018, e.g. MonthFull DD, YYYY

Time Format Used Around the World



Formats for Working with Times

Format	Description	Example
%S	Second as integer (00–61)	00
%OS	Second as decimal number (00-60.99)	00
%M	Minute as decimal number (00–59)	44
%H	Hours as decimal number (00–23)	12
%I	Hours as decimal number (01–12)	12
%p	AM/PM indicator in the locale	PM
%z	Signed offset in hours and minutes from UTC	-0600
%Z	Time zone abbreviation as a character string	CST

For more, see `?strptime`

Formating Non-Standard Times

```
(h_m = as.POSIXct("11:38",
                  format = "%H:%M"))
```

```
## [1] "2018-12-10 11:38:00 CST"
```

```
(h_am = as.POSIXct("11 AM",
                   format = "%I %p"))
```

```
## [1] "2018-12-10 11:00:00 CST"
```

```
(h_m_s_z = as.POSIXct("11:38:22", # Chop off the TZ
                      format = "%H:%M:%S",
                      tz = "America/New_York"))
```

```
## [1] "2018-12-10 11:38:22 EST"
```

Your Turn

How would you convert. . .

- 12:30 PM, e.g. HH:MM A/PM
- 38:22, e.g. MM:SS

Time Zone Notes

- *R* makes use of time zones via `tz` parameter.
- The accepted values of `tz` depend on the location.
  - `CST` is given with `"CST6CDT"` or `"America/Chicago"`
- For supported locations and time zones use:
  - In R: `OlsonNames()`
  - Alternatively, try in R: `system("cat $R_HOME/share/zoneinfo/zone.tab")`
- These locations are given by Internet Assigned Numbers Authority (IANA)
  - List of tz database time zones (Wikipedia)
  - IANA TZ Data (2016e)

Specifics on POSIXct

- POSIXct: Stores time as seconds since UNIX epoch on 1970-01-01 00:00:00
  - Unix & tidyverse preferred format.

```
# POSIXct output
(origin = as.POSIXct("1970-01-01 00:00:00",
                     format ="%Y-%m-%d %H:%M:%S",
                     tz = "UTC"))
```

```
## [1] "1970-01-01 UTC"
```

```
as.numeric(origin)      # At epoch
```

```
## [1] 0
```

```
as.numeric(Sys.time()) # Right now
```

```
## [1] 1544467441
```

On the Agenda

The "other" time object: POSIXlt

- POSIXlt: Stores a list of day, month, year, hour, minute, second, and so on.
  - It is **slower** than POSIXct and has **zero support** in the tidyverse.
  - **Warning:** POSIXlt will be returned if you use strptime()
  - Always convert POSIXlt to POSIXct using as.POSIXct()!!!

```
# POSIXlt output
posixlt = as.POSIXlt(Sys.time(),
                     format ="%Y-%m-%d %H:%M:%S",
                     tz = "America/Chicago")

# Convert to POISXct
posixct = as.POSIXct(posixlt)
posixct
```

```
## [1] "2018-12-10 12:44:01 CST"
```

POSIXlt - List Values

```
posixlt$sec   # Seconds 0-61
```

```
## [1] 1.007323
```

```
posixlt$min   # Minutes 0-59
```

```
## [1] 44
```

```
posixlt$hour  # Hour 0-23
```

```
## [1] 12
```

```
posixlt$mday  # Day of the Month 1-31
```

```
## [1] 10
```

```
posixlt$mon   # Months after the first of the year 0-11
```

```
## [1] 11
```

```
posixlt$year  # Years since 1900.
```

```
## [1] 118
```

anytime

- anytime by Dirk Eddelbuettel seeks to solve the need of remembering date and time formats.
- Main advantage: Only one function anytime() that autodetects the appropriate format and imports it correctly as a POSIXct object.

```
library(anytime)
Sys.setenv(TZ=anytime:::getTZ()) ## helper function to try to get TZ

anytime(c("2017-Jul-10 10:11:12", "Jul/10/2017 10:11:12", "Jul-10-2017 10:11:12"))
```

```
## [1] "2017-07-10 10:11:12 CDT" "2017-07-10 10:11:12 CDT"
## [3] "2017-07-10 10:11:12 CDT"
```

```
anytime(c("Mon Jul 10 10:11:12 2016", "Mon Jul 10 10:11:12.345678 2017"))
```

```
## [1] "2016-07-10 10:11:12 CDT" "2017-07-10 10:11:12 CDT"
```

lubridate - Dates Made Easy

- lubridate by Garret Grolemund, Hadley Wickham, and Gang contains **many** helper functions to write the correct parse syntax e.g.

```
library(lubridate)
ymd("20170710")
```

```
## [1] "2017-07-10"
```

```
interval(mdy("07-10-2017"), dmy("10/07/2017"))
```

```
## [1] 2017-07-10 UTC--2017-07-10 UTC
```

For more, please read the Lubridate vignette on the CRAN

Summary

- To analyze time, you must have it in a `POSIXct` object
  - Avoid `POSIXlt` like the plague.
- Date and Timestamps differ greatly around the world.
- Lots of options outside of base R functions exist.