Dates and times with lubridate :: CHEAT SHEET

Date-times

2017-11-28 12:00:00
A date-time is a point on the timeline, stored as the number of seconds since 1970-01-01 00:00:00 UTC

dt <- as_datetime(1511870400) # "2017-11-28 12:00:00 UTC"

2017-11-28
A date is a day stored as the number of days since 1970-01-01

d <- as_date(17498) # "2017-11-28"

t <- hms::as.hms(85) # 00:01:25

2017-11-28 12:00:00
An hms is a time stored as the number of seconds since 00:00:00

time_decimal(2017.5)
time_decimal(2017.5)

PARSE DATE-TIMES (Convert strings or numbers to date-times)

1. Identify the order of the year (y), month (m), day (d), hour (h), minute (m) and second (s) elements in your data.
2. Use the function below whose name replicates the order. Each accepts a wide variety of input formats.

2017-11-28T14:02:00
2017-22-12 10:00:00
11/28/2017 1:02:03
1 Jan 2017 23:59:59
20170131
July 4th, 2000
4th of July '99
2001 Q3
2:01

GET AND SET COMPONENTS

Use an accessor function to get a component. Assign into an accessor function to change a component in place.

day(x) Date component. date(dt)
year(x) Year. year(dt)

2018-01-31 11:59:59
2018-01-31 11:59:59

2018-01-31
2018-01-31
2018-01-31
2018-01-31
2018-01-31
2018-01-31

Stamp Date-times

stamp() Derive a template from an example string and return a new function that will apply the template to date-times. Also stamp_date() and stamp_time().

1. Derive a template, create a function sf <- stamp("Created Sunday, Jan 17, 1999 3:34")
2. Apply the template to dates sf(ymd("2010-04-05"))

Time Zones

R recognizes ~600 time zones. Each encodes the time zone, Daylight Savings Time, and historical calendar variations for an area. R assigns one time zone per vector.

Use the UTC time zone to avoid Daylight Savings.

OlsonNames() Returns a list of valid time zone names. OlsonNames()

with_tz(time, tzname = "") Get the same date-time in a new time zone (a new clock time).
with_tzdt("US/Pacific")

force_tz(time, tzname = "") Get the same clock time in a new time zone (a new date-time).
force_tz(dt, "US/Pacific")

Round Date-times

floor_date(x, unit = "second") Round down to nearest unit.
floor_date(dt, unit = "month")
round_date(x, unit = "second") Round to nearest unit.
round_date(dt, unit = "month")

ceiling_date(x, unit = "second", change_on_boundary = NULL) Round up to nearest unit.
ceiling_date(dt, unit = "month")

rollback(dates, roll_to_first = FALSE, preserve_hms = TRUE) Roll back to last day of previous month. rollback(dt)
Math with Date-times — Lubridate provides three classes of timespans to facilitate math with dates and date-times

Math with date-times relies on the timeline, which behaves inconsistently. Consider how the timeline behaves during:

- A normal day
- The start of daylight savings (spring forward)
- The end of daylight savings (fall back)
- Leap years and leap seconds

PERIODS
Add or subtract periods to model events that happen at specific clock times, like the NYSE opening bell.

Make a period with the name of a time unit pluralized, e.g.

\[ p \leftarrow \text{months}(3) + \text{days}(12) \]

Periods track changes in clock times, which ignore time line irregularities.

- nor + minutes(90)
- gap + minutes(90)
- lap + minutes(90)
- leap + years(1)

DURATIONS
Add or subtract durations to model physical processes, like battery life. Durations are stored as seconds, the only time unit with a consistent length. Difftimes are a class of durations found in base R.

Make a duration with the name of a period prefixed with a \( d \), e.g.

\[ dd \leftarrow \text{dyears}(14) \]

Durations track the passage of physical time, which deviates from clock time when irregularities occur.

- nor + dminutes(90)
- gap + dminutes(90)
- lap + dminutes(90)
- leap + dyears(1)

INTERVALS
Represent specific intervals of the timeline, bounded by start and end dates-times.

Make an interval with the name of a period prefixed with an \( i \), e.g.

\[ i \leftarrow \text{interval}(\text{ymd}("2017-01-01"), \text{ymd}("2017-01-02")) \]

Intervals represent specific intervals of the timeline.

- interval(nor, nor + minutes(90))
- interval(gap, gap + minutes(90))
- interval(lap, lap + minutes(90))
- interval(leap, leap + years(1))

Durations and intervals allow you to:

- Performing calculations with period and duration objects
- Modeling physical processes
- Tracking changes in clock times
- Representing specific intervals of the timeline
- Performing calculations with interval objects

Not all years are 365 days due to leap days.
Not all minutes are 60 seconds due to leap seconds.

It is possible to create an imaginary date by adding months, e.g. February 31st

\[ \text{jan31} \leftarrow \text{ymd}(20180131) \]

\[ \text{jan31} + \text{months}(1) \]

### Examples

- \[ \text{add_with_rollback}(c(1, 2), \text{roll_to_first} = \text{TRUE}) \]
- \[ \text{add_with_rollback}(\text{jan31}, \text{months}(1), \text{roll_to_first} = \text{TRUE}) \]

### Dates

- Starts June 15, 2017
- Ends June 30, 2017
- First day = June 15, 2017
- Last day = June 30, 2017

### Duration

- Starts June 15, 2017, 05:04 PM
- Ends June 22, 2017, 05:04 PM
- First day = June 15, 2017
- Last day = June 22, 2017

### Interval

- Starts June 15, 2017
- Ends June 16, 2017
- First day = June 15, 2017
- Last day = June 16, 2017

### Notes

- Includes leap days
- Excludes leap seconds

---

RStudio® is a trademark of RStudio, Inc. • CC BY SA RStudio • info@rstudio.com • 844-448-1212 • rstudio.com • Learn more at lubridate.tidyverse.org • lubridate 1.6.0 • Updated: 2017-12