datacamp

Working with Dates and Times in R

Learn R online at www.DataCamp.com

> Definitions used in this cheat sheet

- **Date:** a day stored as the number of days since 1970-01-01
- **POSIXct:** stores date and time in seconds with the number of seconds beginning at 1 January 1970
- **hms:** a simple class for storing durations or time-of-day values and displaying them in the hh:mm:ss format
- **POSIXlt:** stores date and time information as separate components including seconds, minutes, hours, days, etc
- **Interval:** Intervals represent specific intervals of the timeline, bounded by start and end date-times.
- **Period:** Record the datetime ranges/time span in "human" times, Like years and months
- **Duration:** Record the datetime ranges / time span in seconds
- **Difftime:** The difference between two datetime objects

> ISO 8601 datetimes

ISO 8601 specifies datetimes from the largest to the smallest unit of time. **YYYY-MM-DD HH:MM:SS TZ**
Some of the advantages of ISO 8601 are:

- It avoids ambiguities between MM/DD/YYYY and DD/MM/YYYY formats
- The 4-digit year representation mitigates overflow problems after the year 2099
- Using numeric month values (08 not AUG) makes it language independent, so dates makes sense throughout the world.
- R is optimized for this format, since it makes comparison and sorting easier.

> Loading packages

Except where noted, all functionality is found in the `lubridate` package. Some functionality is also found in the `anytime`, `hms`, and `readr` packages.

```
# Load lubridate
library(lubridate)

# Load the other packages
library(anytime)
library(hms)
library(readr)
```

> Getting the current date

```
# Get the current date with today()
today() # "2022-11-11"

# Get the current datetime including timezone with now()
now() # "2022-11-11 08:52:19 EST"
```

> Reading datetime data from CSV

```
# The following uses the readr package

# Read data from a CSV file with read_csv()
read_csv(filename,
        col_types = cols(
            # Specify date column with col_date()
            date_col = col_date("%m/%d/%Y")
            # Specify datetime column with col_datetime()
            datetime_col = col_datetime("%m/%d/%Y %I:%M:%S %p")
            # Specify time column with col_time()
            time_col = col_time("%I:%M:%S %p")
            )
         )
```

> Parsing dates, datetimes and times

Automatic parsing

```
# The following uses the anytime package

# Automatically parse dates in multiple formats with anydate()
anydate(c("Jun 16, 1915", "18 October 1919")) # "1915-06-16" "1919-10-18"

# Automatically parse datetimes in multiple formats with anytime()
anytime(c("22 Nov 1963 13:30", "September 15 1901 02:15"), tz = "EST") # "1963-11-22 13:30:00
EST" "1901-09-15 02:15:00 EST"
```

Manual parsing

```
# Parse dates in year, month, day format with ymd()
ymd("1759 09 22") # "1759-09-22"

# Parse dates in month, day, year format with mdy()
mdy("05-12-1820") # "1820-05-12"

# Parse dates in day, month, year format with dmy()
dmy("01/09/1835") # "1835-09-01"

# Parse datetimes in ISO format
ymd_hms("1972-06-30 23:59:59") # "1972-06-30 23:59:59 UTC"

# Parse datetimes in a single format with fast_strptime()
fast_strptime("January 1, 1924", "%B %d, %Y") # "1924-01-01 UTC"

# Parsing datetimes in multiple specified formats with parse_date_time()
parse_date_time(c("Jun 16, 1915", "18 October 1919"),
                c("%b %d, %Y", "%d %B %Y")) # Returns "1915-06-16 UTC" "1919-10-18 UTC"
```

Parsing times

```
# The following uses the hms package

# Parse times without dates
hms(56,12,15) # Returns 15:12:56
```

> Making dates and datetimes from components

```
# Make dates from components with make_date()
make_date(1777, 4, 30) # "1777-04-30"

# Make datetimes from components with make_datetime()
make_datetime(1945, 6, 16, 05, 29, 21, tz = "US/Mountain") # "1945-06-16 05:29:21 MWT"
```

Extracting components

```
# Extract the year from a date or datetime with year()
year("1923-12-11") # 1923

# Extract the day of the year from a date or datetime with yday()
yday("1900-10-14") # 287

# Extract the month or month name from a date or datetime with month()
month("1857-03-27", label = TRUE) # Mar

# Extract the day of the week from a date or datetime with wday()
wday("1890-02-17", label = TRUE) # Mon
```

> Time zones

The functions below are available in base R.

```
# Get the current timezone
Sys.timezone() # "Asia/Kuala_Lumpur"

# List all known timezones
OlsonNames() # "Africa/Abidjan" ... "Zulu"
```

```
# Specify a datetime that has a location-based timezone
ymd_hms("1915-04-25 12:00:00", tz = "Australia/Eucla") # "1915-04-25 12:00:00 +0845"

# Specify a datetime that has a UTC offset timezone
ymd_hms("1915-04-25 12:00:00 +08:45") 3 "1915-04-25 03:15:00 UTC"

# Use a different timezone for a datetime with with_tz()
with_tz(ymd_hms("1915-04-25 09:00:00", tz = "Asia/Kuala_Lumpur"), "America/Chicago")
# Returns "1915-04-24 20:00:00 CDT"

# Override the timezone for a datetime with force_tz()
force_tz(ymd_hms("1915-04-25 09:00:00", tz = "Asia/Kuala_Lumpur"), "America/Chicago")
# Returns "1915-04-25 09:00:00 CDT"
```

> Time intervals

```
# Some points in time
start_of_time1 <- ymd("1970-01-01")
end_of_time1 <- ymd("2012-12-21")
start_of_time2 <- ymd("2001-01-01")
end_of_time2 <- ymd("2019-12-12")

# Specify the interval between two datetimes with interval()
intrvl1 <- interval(start_of_time1, end_of_time1) # 1970-01-01 UTC--2012-12-21 UTC

# Determine the length of an interval in seconds with int_length()
int_length(intrvl1) # 1356048000

# Determine the overlap between two intervals with intersect()
intrvl2 <- interval(start_of_time2, end_of_time2)
intersect(intrvl1, intrvl2) # 2001-01-01 UTC--2012-12-21 UTC
```

Periods and durations

```
# Define a period in years
years(2) # "2y 0m 0d 0H 0M 0S"

# Define a duration in years
dyears(2) # "63115200s (~2 years)"

# Intervals for a leap year and non-leap year
leap <- interval("2020-01-01", "2021-01-01")
non_leap <- interval("2021-01-01", "2022-01-01")

# Convert an interval to a period with as.period()
as.period(leap) # "1y 0m 0d 0H 0M 0S"
as.period(non_leap) # "1y 0m 0d 0H 0M 0S"

# Convert an interval to a duration with as.duration()
as.duration(leap) # "31622400s (~1 years)"
as.duration(non_leap) # "31536000s (~52.14 weeks)"
```

Date arithmetic

```
# Subtract a historical date from today
today() - ymd("2000-02-29") # Time difference of 8291 days

# Start with a day before a timezone change and add a period of one day
ymd("2022-11-06", tz = "America/New_York") + days(1) # "2022-11-07 EST"

# Start with a day before a timezone change and add a duration of one day
ymd("2022-11-06", tz = "America/New_York") + ddays(1) # "2022-11-06 23:00:00 EST"
```

Rounding dates

```
# Round dates to the nearest time unit
round_date(ymd("2004-10-04"), "week")

# Round dates to the previous time unit
floor_date(ymd("2004-10-04"), "week")

# Round dates to the next time unit
ceiling_date(ymd("2004-10-04"), "week")
```