R - Practice 04 - v1.1

Sangkon Han(sangkon@pusan.ac.kr)

2023-09-11

Contents

Data Wrangle: dates / times (lubridate & hms)

Create dates / times

```
# Basic objects
d <- as_date(18992) # date
t <- as_hms(120) # time
dt <- as_datetime(1640952000) # datetime
```

```
# Parsing date/time: string or number converson...
ymd_hms("2021-12-31 12:00:00") # %>% class()
```

```
## [1] "2021-12-31 12:00:00 UTC"
```

```
ymd_hm("2021-12-31 12:00")
```

```
## [1] "2021-12-31 12:00:00 UTC"
```

```
ymd_h("2021-12-31 12")
```

```
## [1] "2021-12-31 12:00:00 UTC"
```

```
ydm_hms("2021-31-12 11:30:00")
```

```
## [1] "2021-12-31 11:30:00 UTC"
```

```
ydm_hm("2021-31-12 11:30")
```

```
## [1] "2021-12-31 11:30:00 UTC"
```

```
ydm_h("2021-31-12 11")
```

```
## [1] "2021-12-31 11:00:00 UTC"
```

```
mdy_hms("12/31/2021 3:05:05")
```

```
## [1] "2021-12-31 03:05:05 UTC"
```

```
dmy_hms("31 Dec 2021 22/15:00")
```

```
## [1] "2021-12-31 22:15:00 UTC"
```

```
ymd("20211231")
```

```
## [1] "2021-12-31"
```

```
mdy("December 13st 2021")
```

```
## [1] "2021-12-13"
```

```
dmy("31st of December 21")
```

```
## [1] "2021-12-31"
```

```
yq("2021: Q4")
```

```
## [1] "2021-10-01"
```

```
hms::hms(seconds = 5, minutes = 1, hours = 0)
```

```
## 00:01:05
```

```
lubridate::hms("00:01:05")
```

```
## [1] "1M 5S"
```

```
# date_decimal() - Parse date stored as decimal number
d <- seq(2021,2022,0.25)
date_decimal(d)
```

```
## [1] "2021-01-01 00:00:00 UTC" "2021-04-02 06:00:00 UTC"
## [3] "2021-07-02 12:00:00 UTC" "2021-10-01 18:00:00 UTC"
## [5] "2022-01-01 00:00:00 UTC"
```

```
# fast_strptime() - Parse datetime
fast_strptime(x = "2021-12-31 12:00:00", format = "%Y-%m-%d %H:%M:%S")
```

```
## [1] "2021-12-31 12:00:00 UTC"
```

```
# parse_date_time() - Easier parse datetime
parse_date_time("2021-12-31 12:00:00", "ymd HMS")
```

```
## [1] "2021-12-31 12:00:00 UTC"
```

```
# Create date/time from individual components
flights %>% head()
```

```
## # A tibble: 6 x 19
##    year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time
##   <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>
## 1  2013     1     1      517            515         2      830            819
## 2  2013     1     1      533            529         4      850            830
## 3  2013     1     1      542            540         2      923            850
## 4  2013     1     1      544            545        -1     1004           1022
## 5  2013     1     1      554            600        -6      812            837
## 6  2013     1     1      554            558        -4      740            728
## # i 11 more variables: arr_delay <dbl>, carrier <chr>, flight <int>,
## #   tailnum <chr>, origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>,
## #   hour <dbl>, minute <dbl>, time_hour <dttm>
```

```
# Create datetime and date column using other components
flights %>%
  select(year, month, day, hour, minute) %>%  # components
  mutate(datetime = make_datetime(year, month, day, hour, minute), # datetime create
         date = make_date(year, month, day)) %>%  # date create
  head()
```

```
## # A tibble: 6 x 7
##    year month   day  hour minute datetime            date
##   <int> <int> <int> <dbl>  <dbl> <dttm>              <date>
## 1  2013     1     1     5     15 2013-01-01 05:15:00 2013-01-01
## 2  2013     1     1     5     29 2013-01-01 05:29:00 2013-01-01
## 3  2013     1     1     5     40 2013-01-01 05:40:00 2013-01-01
## 4  2013     1     1     5     45 2013-01-01 05:45:00 2013-01-01
## 5  2013     1     1     6      0 2013-01-01 06:00:00 2013-01-01
## 6  2013     1     1     5     58 2013-01-01 05:58:00 2013-01-01
```

```
# Create date/time from existing objects
# Current timestamp and todays date
now()
```

```
## [1] "2023-09-11 00:23:31 KST"
```

```
today()
```

```
## [1] "2023-09-11"
```

```
# Convert between datetime and date
as_date(now())
```

```
## [1] "2023-09-11"
```

```
as_datetime(today())
```

```
## [1] "2023-09-11 UTC"
```

```
# Different codes for datetimes components:
# Code  Value
#--------------------------------------------#
# %d    Day of the month (decimal number)
# %a    Abbreviated weekday
# %m    Month (decimal number)
# %A    Full weekday
# %b    Month (abbreviated)
# %I    Decimal hour (12 hour)
# %B    Month (full name)
# %j    Decimal day of the year
# %y    Year (2 digits)
# %w    Decimal Weekday (0=Sunday)
# %Y    Year (4 digits)
# %W    Decimal week of the year (starting on Monday)
# %H    Decimal hour (24 hour)
# %p    Locale-specific AM/PM
# %M    Decimal minute
# %x    Locale-specific Date
# %S    Decimal second
# %X    Locale-specific Time
```

Components

```r
# Extract different components of current time stamp
dt <- now()
dt
```

```
## [1] "2023-09-11 00:23:31 KST"
```

```r
# Extract each piece of datetime
year(dt)
```

```
## [1] 2023
```

```r
month(dt)
```

```
## [1] 9
```

```r
day(dt)
```

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

```r
hour(dt)
```

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

```r
minute(dt)
```

```
## [1] 23
```

```r
second(dt)
```

```
## [1] 31.2234
```

```r
# Some additional components
isoyear(dt)
```

```
## [1] 2023
```

```r
epiyear(dt)
```

```
## [1] 2023
```

```r
wday(dt)
```

```
## [1] 2
```

```r
qday(dt)
```

```
## [1] 73
```

```r
week(dt)
```

```
## [1] 37
```

```r
isoweek(dt)
```

```
## [1] 37
```

```r
epiweek(dt)
```

```
## [1] 37
```

```
quarter(dt)
```

```
## [1] 3
```

```
semester(dt)
```

```
## [1] 2
```

```
# Logicals
am(dt)
```

```
## [1] TRUE
```

```
pm(dt)
```

```
## [1] FALSE
```

```
dst(dt)
```

```
## [1] FALSE
```

```
leap_year(dt)
```

```
## [1] FALSE
```

```
# Store value of a component into column
flights %>%
  select(year, month, day, hour, minute) %>%  # components
  mutate(datetime = make_datetime(year, month, day, hour, minute)) %>%  # datetime create
  # extract week day, week and quarter
  mutate(wday = wday(datetime),
         week = week(datetime),
         Q = quarter(datetime)) %>%
  head()
```

```
## # A tibble: 6 x 9
##    year month   day  hour minute datetime             wday  week     Q
##   <int> <int> <int> <dbl>  <dbl> <dttm>              <dbl> <dbl> <int>
## 1  2013     1     1     5     15 2013-01-01 05:15:00     3     1     1
## 2  2013     1     1     5     29 2013-01-01 05:29:00     3     1     1
## 3  2013     1     1     5     40 2013-01-01 05:40:00     3     1     1
## 4  2013     1     1     5     45 2013-01-01 05:45:00     3     1     1
## 5  2013     1     1     6      0 2013-01-01 06:00:00     3     1     1
## 6  2013     1     1     5     58 2013-01-01 05:58:00     3     1     1
```

Rounding values & setting component

```
# Rounding dates per month level
d <- today()
d
```

```
## [1] "2023-09-11"
```

```
floor_date(d, unit = "month")   # round down to previous month
```

```
## [1] "2023-09-01"
```

```
ceiling_date(d, unit = "month") # round up to next month
```

```
## [1] "2023-10-01"
```

```r
round_date(d, unit = "month")   # mathematical rules for rounding
```

```
## [1] "2023-09-01"
```

```r
rollback(d) # rollback to last day of previous month
```

```
## [1] "2023-08-31"
```

```r
# It also works for other units
floor_date(d, unit = "year")
```

```
## [1] "2023-01-01"
```

```r
ceiling_date(dt, unit = "day")
```

```
## [1] "2023-09-12 KST"
```

```r
round_date(dt, unit = "minute")
```

```
## [1] "2023-09-11 00:24:00 KST"
```

```r
# Updating components
# Update each component  by assigning new values
dt
```

```
## [1] "2023-09-11 00:23:31 KST"
```

```r
year(dt) <- 2022
month(dt) <- 12
day(dt) <- 31
hour(dt) <- 23
minute(dt) <- 59
second(dt) <- 59

# Update all components in one take with update()
update(dt, year = 2022, month = 12, day = 31, hour = 23, minute = 59, second = 59)
```

```
## [1] "2022-12-31 23:59:59 KST"
```

```r
# Too great values rollback!
update(dt, month = 13) # 12 + 1 months = 1 year + 1 month
```

```
## [1] "2023-01-31 23:59:59 KST"
```

```r
update(dt, hour = 25)  # 24 + 1 hour = 1 day  + 1 hour
```

```
## [1] "2023-01-01 01:59:59 KST"
```

Date-times arithmetics and durations

```r
# Some basic date/time arithmetics
today <- Sys.Date()
today
```

```
## [1] "2023-09-11"
```

```r
today + 1 # tomorrow
```

```
## [1] "2023-09-12"
```

```r
today - 1 # yesterday
```

```
## [1] "2023-09-10"
```

```r
now <- Sys.time()
now
```

```
## [1] "2023-09-11 00:23:31 KST"
```

```r
now + 3600 # after 1 hour
```

```
## [1] "2023-09-11 01:23:31 KST"
```

```r
now - 3600 # before 1 hour
```

```
## [1] "2023-09-10 23:23:31 KST"
```

```r
# How old are you?
birth_date <- ymd("1987-05-28")
age <- today - birth_date
age
```

```
## Time difference of 13255 days
```

```r
# Durations
# Convert age to duration
as.duration(age) # in seconds
```

```
## [1] "1145232000s (~36.29 years)"
```

```r
# Durations constructor functions
x <- 1 # number of seconds
dyears(x)
```

```
## [1] "31557600s (~1 years)"
```

```r
dmonths(x)
```

```
## [1] "2629800s (~4.35 weeks)"
```

```r
dweeks(x)
```

```
## [1] "604800s (~1 weeks)"
```

```r
ddays(x)
```

```
## [1] "86400s (~1 days)"
```

```r
dhours(x)
```

```
## [1] "3600s (~1 hours)"
```

```r
dminutes(x)
```

```
## [1] "60s (~1 minutes)"
```

```r
dpicoseconds(x)
```

```
## [1] "1e-12s"
```

```r
is.duration(age)
```

```
## [1] FALSE
```

```r
is.duration(as.duration(age))
```

```
## [1] TRUE
```

```r
# Durations - arithmetics
dseconds(10) + dminutes(1) # addition
```

```
## [1] "70s (~1.17 minutes)"
```

```r
dyears(1) - dweeks(27)      # subtraction
```

```
## [1] "15228000s (~25.18 weeks)"
```

```r
10 * dmonths(1)             # multiplication
```

```
## [1] "26298000s (~43.48 weeks)"
```

```r
# Inconsistent timeline behaviour (durations)
# Daylight Savings Time
dt <- ymd_hms("2016-03-12 13:00:00", tz = "America/New_York")
dt + ddays(1)
```

```
## [1] "2016-03-13 14:00:00 EDT"
```

```r
# Leap year
dt <- ymd_hms("2019-02-28 23:00:00")
dt +  dyears(1)
```

```
## [1] "2020-02-29 05:00:00 UTC"
```

```r
# Periods
# Age as period
as.period(age)
```

```
## [1] "13255d 0H 0M 0S"
```

```r
# Constructor functions
seconds(3600)
```

```
## [1] "3600S"
```

```r
minutes(60)
```

```
## [1] "60M 0S"
```

```r
hours(1)
```

```
## [1] "1H 0M 0S"
```

```r
days(1)
```

```
## [1] "1d 0H 0M 0S"
```

```r
months(12)
```

```
## [1] "12m 0d 0H 0M 0S"
```

```r
weeks(54)
```

```
## [1] "378d 0H 0M 0S"
```

```r
years(1)
```

```
## [1] "1y 0m 0d 0H 0M 0S"
```

```r
period_to_seconds(years(1))
```

```
## [1] 31557600
```

```r
seconds_to_period(3600)
```

```
## [1] "1H 0M 0S"
```

```r
period(3600, units = "second")
```

```
## [1] "3600S"
```

```r
# Periods - arithmetics
seconds(10) + minutes(1) # addition
```

```
## [1] "1M 10S"
```

```r
years(1) - weeks(27)      # subtraction
```

```
## [1] "1y 0m -189d 0H 0M 0S"
```

```r
10 * months(1)            # multiplication
```

```
## [1] "10m 0d 0H 0M 0S"
```

```r
# Inconsistent timeline behaviour (periods)
# Daylight Savings Time
dt <- ymd_hms("2016-03-12 13:00:00", tz = "America/New_York")
dt + days(1)
```

```
## [1] "2016-03-13 13:00:00 EDT"
```

```r
# Leap year
dt <- ymd_hms("2019-02-28 23:00:00")
dt +  years(1)
```

```
## [1] "2020-02-28 23:00:00 UTC"
```

Intervals

```r
# Create an interval
d1 <- ymd("2021-12-30")
d2 <- ymd("2021-12-31")

i1 <- interval(d1, d2)
i2 <- d2 %--% d1
i1
```

```
## [1] 2021-12-30 UTC--2021-12-31 UTC
```

```r
i2
```

```
## [1] 2021-12-31 UTC--2021-12-30 UTC
```

```r
# Extract boundaries
int_start(i1)
```

```
## [1] "2021-12-30 UTC"
```

```r
int_end(i1)
```

```
## [1] "2021-12-31 UTC"
```

```r
# Is time point within given interval
ymd_hms("2021-12-30 01:00:00") %within% i1
```

```
## [1] TRUE
```

```r
ymd_hms("2021-12-29 23:00:00") %within% i1
```

```
## [1] FALSE
```

```r
# Do intervals overlap
int_overlaps(i1, i2)
```

```
## [1] TRUE
```

```r
int_overlaps(i1, ymd("2021-12-28") %--% ymd("2021-12-29"))
```

```
## [1] FALSE
```

```r
# Create intervals form vector of dates
dates <- now() + days(1:365) # one year of dates
int_diff(dates) %>% head()# daily intervals
```

```
## [1] 2023-09-12 00:23:32 KST--2023-09-13 00:23:32 KST
## [2] 2023-09-13 00:23:32 KST--2023-09-14 00:23:32 KST
## [3] 2023-09-14 00:23:32 KST--2023-09-15 00:23:32 KST
## [4] 2023-09-15 00:23:32 KST--2023-09-16 00:23:32 KST
## [5] 2023-09-16 00:23:32 KST--2023-09-17 00:23:32 KST
## [6] 2023-09-17 00:23:32 KST--2023-09-18 00:23:32 KST
```

```r
# Length of an interval / flip interval
i1
```

```
## [1] 2021-12-30 UTC--2021-12-31 UTC
```

```r
int_flip(i1)
```

```
## [1] 2021-12-31 UTC--2021-12-30 UTC
```

```r
int_length(i1)
```

```
## [1] 86400
```

Time zones

```r
# What R sees as your time zone?
Sys.timezone()
```

```
## [1] "Asia/Seoul"
```

```r
# Different time zones
OlsonNames() %>% head()
```

```
## [1] "Africa/Abidjan"     "Africa/Accra"       "Africa/Addis_Ababa"
## [4] "Africa/Algiers"     "Africa/Asmara"      "Africa/Asmera"
```

```r
# How many different time zones
OlsonNames() %>% length()
```

```
## [1] 596
```

```r
# "US" ~ "Europe" included in TZ
OlsonNames() %>% str_subset(pattern = "US") %>% head()
```

```
## [1] "US/Alaska"       "US/Aleutian"     "US/Arizona"      "US/Central"
## [5] "US/East-Indiana" "US/Eastern"
```

```r
OlsonNames() %>% str_subset(pattern = "Europe") %>% head()
```

```
## [1] "Europe/Amsterdam" "Europe/Andorra"   "Europe/Astrakhan" "Europe/Athens"
## [5] "Europe/Belfast"   "Europe/Belgrade"
```