

Data Wrangling with R: Day 2

Dealing with dates with lubridate

Presented by Emi Tanaka

Department of Econometrics and Business Statistics



MONASH University

2nd December 2020 @ Statistical Society of Australia | Zoom

Date in R Base R Part 1

- Dealing with dates alone is relatively straightforward compared to date and time
- Dealing with date *and* time is ... tricky so let's start with dates

```
Sys.Date() # System Date, gets the date when the command is run
```

```
## [1] "2020-12-02"
```

- Dates in R have class **Date**  even though it looks like character 

```
class(Sys.Date())
```

```
## [1] "Date"
```

- It's actually a numerical value under the hood, what is this number? 

```
unclass(Sys.Date())
```

```
## [1] 18598
```

Date in R

Base R Part 2

- ⓘ 1st January 1970 is a special reference point
- Let's have a look at the numerical value under the hood of Date objects

```
unclass(as.Date("1970/01/02"))
```

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

```
unclass(as.Date("1969/12/31"))
```

```
## [1] -1
```

- Yup, the number under the hood is the number of days after (if positive) or before (if negative) 1st January 1970
- And yes, you can use `as.Date` to convert objects to Date 📅

Date in R Base R Part 3

- Dates do no have to be in the format of "YYYY/MM/DD" (in fact, there are many format in the wild)
- If it has a different format, then you can use the conversion specification with a "%" symbol followed by a single letter note quite regex, but like it

```
as.Date("Xmas is 25 December 2020", format = "Xmas is %d %B %Y")  
## [1] "2020-12-25"
```

- You can find some widely used conversion specification in documentation at `?strptime` but some *depends on your operating system*
- Below are some common ones:
 - %b abbreviated month
 - %e day of the month (01, 02, ..., 31)
 - %y year without century (00-99)
 - %B full month



System locale

- "aralık" is December in Turkey

```
as.Date("Xmas is 25 aralık 2020", format = "Xmas is %d %B %Y")  
## [1] NA
```

- Let's temporary set our system locale to Turkey

```
Sys.setlocale("LC_TIME", "tr_TR.UTF-8") # temporary set to Turkey locale  
as.Date("Xmas is 25 aralık 2020", format = "Xmas is %d %B %Y")  
## [1] "2020-12-25"
```

(And set it back to English again) "UTF-8" might only for Unix and Linux systems

```
Sys.setlocale("LC_TIME", "en_AU.UTF-8")
```

Date and Time in R

Base R Part 1

- two main date-time classes in R: **POSIXct** and **POSIXlt** avoid using POSIXlt if possible
- POSIX stands for Portable Operating System Interface
- ct stands for calendar time

```
as.POSIXct("2020-12-02 13:00", format = "%Y-%m-%e %H:%M")  
## [1] "2020-12-02 13:00:00 AEDT"  
  
unclass(as.POSIXct("2020-12-02 13:00", format = "%Y-%m-%e %H:%M"))  
## [1] 1606874400  
## attr(,"tzone")  
## [1] ""
```

- ⓘ 1970/01/01 00:00:00 UTC is a special reference point called *Unix epoch* and the above number is the number of seconds after Unix epoch

Date and Time in R

Base R Part 2

- POSIXlt seems like it's the same as POSIXct

```
as.POSIXlt("2020-12-02 13:00", format = "%Y-%m-%e %H:%M")  
## [1] "2020-12-02 13:00:00 AEDT"
```

- But under the hood, it's a list of time attributes

```
unclass(as.POSIXlt("2020-12-02 13:00", format = "%Y-%m-%e %H:%M"))  
## $sec  
## [1] 0  
##  
## $min  
## [1] 0  
##  
## $hour  
## [1] 13
```

- lubridate is a package in Tidyverse but not part of the core pacakges
- This means that you need to `library(lubridate)` explicitly if you need to use it

Time zone

```
melb <- as.POSIXct("2020-12-02 13:00", format = "%Y-%m-%e %H:%M",  
tz = "Australia/Melbourne")
```

```
perth <- as.POSIXct("2020-12-02 13:00", format = "%Y-%m-%e %H:%M",  
tz = "Australia/Perth")
```

```
melb - perth
```

Time difference of -3 hours

- You can find the names of the time zones using **OlsonNames()**
- If you want to know which time zone your system is using:

```
Sys.timezone()
```

[1] "Australia/Melbourne"

Working with lubridate

Date in R lubridate

- Remember, lubridate isn't part of core tidyverse so you have to load it up explicitly

```
library(lubridate)
```

- To convert string to a Date, you can use ymd and friends. E.g.

```
ymd("2012 Dec 30th")  
## [1] "2012-12-30"  
  
mdy("01/30 99")  
## [1] "1999-01-30"  
  
dmy("1st January 2015")  
## [1] "2015-01-01"
```



You might have guessed it but:
y = year, m = month, and d = day.

The order determines the
expected order of its appearance
in the string

Date and time in R `lubridate`

- To convert string to POSIXct, you can use `ymd_hms` and friends

```
ymd_hms("20140101 201001", tz = "Australia/Melbourne")
```

```
## [1] "2014-01-01 20:10:01 AEDT"
```

```
mdy_h("09/09/2010 4PM")
```

```
## [1] "2010-09-09 16:00:00 UTC"
```

```
ydm_hm("Today is not 2009 9th Sep 4:30PM")
```

```
## [1] "2009-09-09 16:30:00 UTC"
```

```
ydm_hms("19 9 July | 4:30:03.34343")
```

```
## [1] "2019-07-09 04:30:03 UTC"
```



h = hour, m = minute, and s = second.

It's remarkably clever!

The time has to be after date though.

Conversion to date and time `lubridate`

Making Date from individual date components:

```
make_date(year = 2018,  
          month = 8,  
          day = 3)  
  
## [1] "2018-08-03"
```

Making POSIXct from individual components:

```
make_datetime(year = 2018,  
             month = 8,  
             day = 3,  
             hour = 10,  
             min = 3,  
             sec = 30)  
  
## [1] "2018-08-03 10:03:30 UTC"
```

Extracting date or time components `lubridate`

```
t1 <- ymd_hms("20101010 13:30:30")  
  
month(t1, label = TRUE)  
## [1] Oct  
## 12 Levels: Jan < Feb < Mar < Apr < May < Jun < Jul < Aug < Sep < ... < Dec  
  
year(t1)  
## [1] 2010  
  
month(t1)  
## [1] 10  
  
day(t1)  
## [1] 10  
  
hour(t1)  
## [1] 13  
  
minute(t1)  
## [1] 30  
  
second(t1)  
## [1] 30  
  
yday(t1)  
## [1] 283  
  
mday(t1)  
## [1] 10  
  
wday(t1)  
## [1] 1
```

Date and time modifiers

```
month(t1) <- 3  
t1  
  
## [1] "2010-03-10 13:30:30 UTC"  
  
mday(t1) <- 20  
t1  
  
## [1] "2010-03-20 13:30:30 UTC"  
  
with_tz(t1, "Australia/Perth")  
  
## [1] "2010-03-20 21:30:30 AWST"
```

Durations lubridate

- Duration is a special class in lubridate
- Some convenient constructors for Duration are:

```
dyears(1)
```

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

```
dweeks(10)
```

```
## [1] "6048000s (~10 weeks)"
```

```
ddays(4)
```

```
## [1] "345600s (~4 days)"
```

```
dhours(3)
```

```
## [1] "10800s (~3 hours)"
```

Maths with Durations `lubridate`

```
ddays(4) + dweeks(1)

## [1] "950400s (~1.57 weeks)"

ymd("2013-01-01") + ddays(5)

## [1] "2013-01-06"

ymd_hms("2020-10-1 2:00:00", tz = "Australia/Melbourne") + ddays(1)

## [1] "2020-10-02 02:00:00 AEST"
```

- What happened below?

```
ymd_hms("2020-10-4 1:00:00", tz = "Australia/Melbourne") + dhours(1)

## [1] "2020-10-04 03:00:00 AEDT"
```

- Day light saving started at Sun 4th Oct 2020 2AM in Melbourne

Period lubridate

- Period is a special class in lubridate
- Constructors for Period are like for Duration but without the prefix "d":

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

```
weeks(10)  
## [1] "70d 0H 0M 0S"
```

```
days(4)  
## [1] "4d 0H 0M 0S"
```

```
hours(3)  
## [1] "3H 0M 0S"
```

Maths with Period lubridate

```
days(4) + weeks(1)

## [1] "11d 0H 0M 0S"

ymd("2013-01-01") + days(5)

## [1] "2013-01-06"

ymd_hms("2020-10-1 2:00:00", tz = "Australia/Melbourne") + days(1)

## [1] "2020-10-02 02:00:00 AEST"

ymd_hms("2020-10-4 1:00:00", tz = "Australia/Melbourne") + hours(1)

## [1] NA

ymd_hms("2020-10-4 1:00:00", tz = "Australia/Melbourne") + hours(2)

## [1] "2020-10-04 03:00:00 AEDT"
```

**</> If you installed the `dwexercise` package,
run below in your R console**

```
learnr::run_tutorial("day2-exercise-03", package = "dwexercise")
```

 **If the above doesn't work for you, go [here](#).**
 **Questions or issues, let us know!**

15:00

Session Information

```
devtools::session_info()
```

```
## - Session info -----
##   setting  value
##   version  R version 4.0.1 (2020-06-06)
##   os        macOS Catalina 10.15.7
##   system   x86_64, darwin17.0
##   ui        X11
##   language (EN)
##   collate  en_AU.UTF-8
##   ctype    en_AU.UTF-8
##   tz       Australia/Melbourne
##   date     2020-12-02
##
## - Packages -----
##   package * version  date     lib source
##   anicon      0.1.0    2020-06-21 [1] Github (emitanaka/anicon@0b756df)
##   assertthat   0.2.1    2019-03-21 [2] CRAN (R 4.0.0)
```

These slides are licensed under

