使用 lubridate 處理日期和時間::備忘表

日期和時間

Date-times

2023-09-07 00:00:00
Date-times是時間軸上的一個點，存儲為自1970-01-01 00:00:00 UTC以來的秒數

dt <- as_datetime(1694044800)
"2023-09-07 00:00:00 UTC"

2023-08-19
date 是一個日期，存儲為自1970-01-01以來的天數

d <- as_date(19588)
"2023-08-19"

12:00:00
Time(hms) 是一個時間格式，存儲為自每天00:00:00以來的秒數

t <- hms::as_hms(85)
00:01:25

解析日期和時間（將字串或數字轉換為日期和時間）

1. 在您的數據中識別年（y）、月（m）、日（d）、小時（h）、分鐘（m）和秒（s）元素的順序。
2. 使用下面的函數，其名稱複製了順序。每個函數都接受一個 tz 參數來設置時區，例如 ymd(x, tz = "UTC")

2017-11-28T14:02:00 — **ymd_hms(), ymd_hm(), ymd_h().** ymd_hms("2017-11-28T14:02:00")

2017-22-12 10:00:00 — **ydm_hms(), ydm_hm(), ydm_h().** ydm_hms("2017-22-12 10:00:00")

11/28/2017 1:02:03 — **mdy_hms(), mdy_hm(), mdy_h().** mdy_hms("11/28/2017 1:02:03")

1 Jan 2017 23:59:59 — **dmy_hms(), dmy_hm(), dmy_h().** dmy_hms("1 Jan 2017 23:59:59")

20170131 — **ymd(), ydm().** ymd(20170131)

July 4th, 2000 — **mdy(), myd().** mdy("July 4th, 2000")

4th of July '99 — **dmy(), dym().** dmy("4th of July '99")

2001: Q3 — **yq()** Q for quarter. yq("2001: Q3")

07-2020 — **my(), ym().** my("07-2020")

2:01 — hms::**hms()** Also lubridate::**hms()**, **hm()** and **ms()**, which return periods.* hms::hms(seconds = 0, minutes = 1, hours = 2)

2017.5 — **date_decimal(**decimal, tz = "UTC"**)** date_decimal(2017.5)

now(tzone = ""**)** Current time in tz (defaults to system tz). now()

today(tzone = ""**)** Current date in a tz (defaults to system tz). today()

fast_strptime() Faster strptime. fast_strptime("9/1/01", "%y/%m/%d")

parse_date_time() Easier strptime. parse_date_time("09-01-01", "ymd")

取得和設置組件

GET AND SET COMPONENTS

使用訪問器函數來獲取組件。將值分配到訪問器函數中以就地更改組件。

```
d ## "2017-11-28"
day(d) ## 28
day(d) <- 1
d ## "2017-11-01"
```

2018-01-31 11:59:59 — date(x) 日期組件. date(dt)

2018-01-31 11:59:59 — year(x) Year(西元年). year(dt)
isoyear(x) The ISO 8601 year.
epiyear(x) Epidemiological year.

2018-01-31 11:59:59 — month(x, label, abbr) Month(月份). month(dt)

2018-01-31 11:59:59 — day(x) Day of month(日期). day(dt)
wday(x, label, abbr) 星期幾.
qday(x) Day of quarter.

2018-01-31 11:59:59 — hour(x) Hour(小時). hour(dt)

2018-01-31 11:59:59 — minute(x) Minutes(分). minute

2018-01-31 11:59:59 — (dt) second(x) Seconds(秒).

2018-01-31 11:59:59 UTC — second(dt) tz(x) Time zone(時區). tz(dt)

week(x)一年中的週數. week(dt)
isoweek() ISO 8601 week.
epiweek() Epidemiological week.

quarter(x) Quarter(季). quarter(dt)

semester(x, with_year = FALSE)
Semester(學期). semester(dt)

am(x) 是在上午嗎? am(dt)

pm(x) 是在下午嗎? pm(dt)

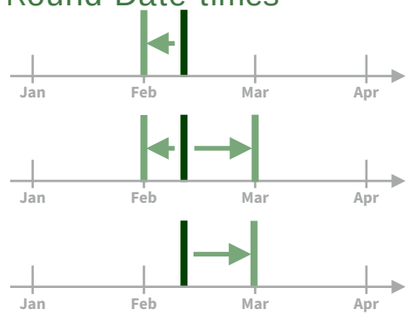
dst(x) 是否是夏令時? ? dst(d)

leap_year(x) 今年是閏年嗎?
leap_year(d)

update(object, ..., simple = FALSE)
update(dt, mday = 2, hour = 1)

四捨五入日期和時間

Round Date-times



floor_date(x, unit = "second") 向下四捨五入到最接近的單位
floor_date(dt, unit = "month")

round_date(x, unit = "second") 四捨五入到最接近的單位.
round_date(dt, unit = "month")

ceiling_date(x, unit = "second", change_on_boundary = NULL) 向上四捨五入到最接近的單位。. ceiling_date(dt, unit = "month")

有效的單位有 second, minute, hour, day, week, month, bimonth, quarter, season, halfyear and year.

rollback(dates, roll_to_first = FALSE, preserve_hms = TRUE) 回到上個月的最後一天. Also rollforward(). rollback(dt)

戳記日期和時間Stamp Date-times

stamp()從示例字符串中派生一個模板，並返回一個新函數，該函數將模板應用於日期和時間。. Also stamp_date() and stamp_time().

1. 生成一個模板，創建一個函數
   sf <- stamp("Created Sunday, Jan 17, 1999
2. 將模板應用於日期
   sf(ymd("2010-04-05"))
   ## [1] "Created Monday, Apr 05, 2010 00:00"

Tip: use a date with day > 12

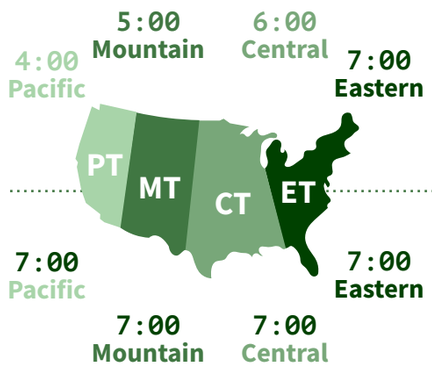
時區Time Zones

R 可識別約 600 個時區。每個都對一個地區的時區、夏令時和歷史日曆變化進行編碼。 R 為每個向量分配一個時區。

使用 UTC 時區以避免夏令時。

OlsonNames() 返回有效時區名稱的列表。

Sys.timezone() 獲取當前時區。



with_tz(time, tzone = "") 獲取新時區中的相同日期和時間（一個新的時鐘時間）. 還有 local_time(dt, tz, units). with_tz(dt, "US/Pacific")

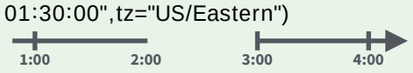
force_tz(time, tzone = "") 在新時區中獲取相同的時鐘時間（一個新的日期和時間）. Also force_tzs(). force_tz(dt, "US/Pacific")

日期和時間進行數學運算 — Lubridate 提供三種時間跨度類別，以便於與日期和日期時間進行數學運算。

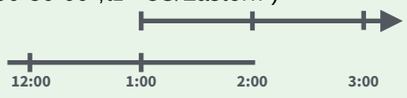
與日期和時間的數學運算依賴於時間軸，其行為不一致。請考慮在以下情況下時間軸的行為：

標準的一天
nor <- ymd_hms("2018-01-01 01:30:00",tz="US/Eastern")

夏令時的開始（向前調整時間）
gap <- ymd_hms("2018-03-11 01:30:00",tz="US/Eastern")



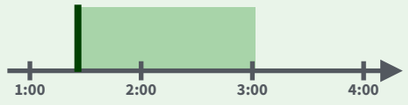
夏令時的結束（向後調整時間）
lap <- ymd_hms("2018-11-04 00:30:00",tz="US/Eastern")



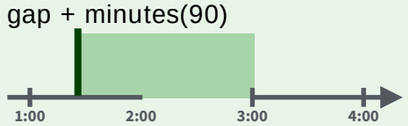
閏年和閏秒
leap <- ymd("2019-03-01")

Periods 週期追踪時鐘時間的變化，忽略時間軸的不規則性。

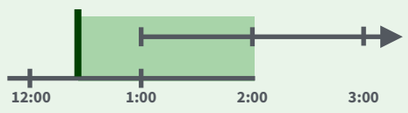
nor + minutes(90)



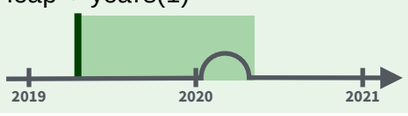
gap + minutes(90)



lap + minutes(90)

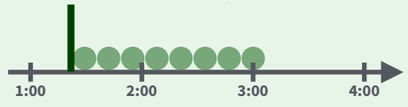


leap + years(1)



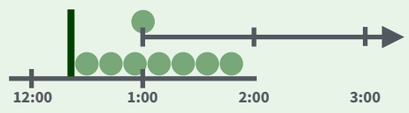
Durations 持續時間追踪物理時間的流逝，當不規則性發生時，它會偏離時鐘時間。

nor + dminutes(90)

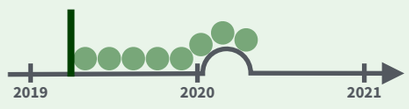


gap + dminutes(90)

lap + dminutes(90)

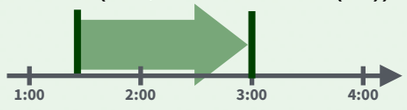


leap + dyears(1)

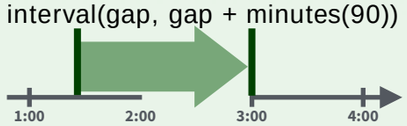


Intervals 時間軸的特定區間，由起始和結束的日期和時間界定

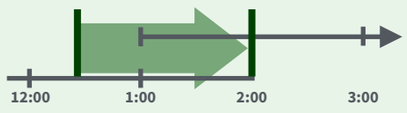
interval(nor, nor + minutes(90))



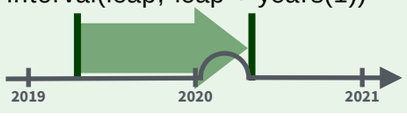
interval(gap, gap + minutes(90))



interval(lap, lap + minutes(90))



interval(leap, leap + years(1))



由於有閏日並非所有年份都是365天。
由於有閏秒並非所有分鐘都是60秒，
通過添加月份，例如2月31日，可以創建一個虛構的日期。 e.g. February 31st

jan31 <- ymd(20180131)
jan31 + months(1)
NA

%m+% and %m-% 將把虛構的日期滾動到上個月的最後一天。

jan31 %m+% months(1)
"2018-02-28"

add_with_rollback(e1, e2, roll_to_first = TRUE) 將把虛構的日期滾動到新月份的第一天.

add_with_rollback(jan31, months(1), roll_to_first = TRUE)
"2018-03-01"

Durations註解:在考慮到如夏令時調整、閏秒等不規則因素時。物理時間是基於物理現象的連續和恆定的流逝，而時鐘時間則可能會受到這些不規則因素的影響。

PERIODS週期

添加或減去週期以模擬在特定時鐘時間發生的事件，例如紐約證券交易所的開市鐘聲。
使用時間單位的複數名稱創建一個週期，例如
p <- months(3) + days(12)
p
"3m 12d 0H 0M 0S"

Number of months | Number of days | etc.

years(x = 1) x years.
months(x) x months.
weeks(x = 1) x weeks.
days(x = 1) x days.
hours(x = 1) x hours.
minutes(x = 1) x minutes.
seconds(x = 1) x seconds.
milliseconds(x = 1) x milliseconds.
microseconds(x = 1) x microseconds
nanoseconds(x = 1) x nanoseconds.
picoseconds(x = 1) x picoseconds.
period(num = NULL, units = "second", ...) 是一個用於自動化的時間段建構函數
period(5, unit = "years")

as.period(x, unit)將時間跨度轉換為時間段，可以選擇性地使用指定的單位進行轉換。 Also is.period(). as.period(p)
period_to_seconds(x) 將時間段轉換為該時間段隱含的「標準」秒數。 Also seconds_to_period().
period_to_seconds(p)

DURATIONS持續時間

新增或減去持續時間以模擬物理過程，如電池壽命。持續時間以秒為單位存儲，這是唯一具有一致長度的時間單位。Difftimes 是在基本 R 中找到的一類持續時間類別。

使用以「d」為前綴的時間段名稱創建一個持續時間， e.g.

dd <- ddays(14)
dd
"1209600s (~2 weeks)"

Exact length in seconds | Equivalent in common units

dyears(x = 1) 31536000x seconds. dmonths(x = 1) 2629800x seconds. dweeks(x = 1) 604800x seconds. ddays (x = 1) 86400x seconds.
dhours(x = 1) 3600x seconds.
dminutes(x = 1) 60x seconds.
dseconds(x = 1) x seconds.
dmilliseconds(x = 1) $x10^{-3}$ seconds.
dmicroseconds(x = 1) $x10^{-6}$ seconds.
dnanoseconds(x = 1) x 10^{-9} seconds.
dpicoseconds(x = 1) x 10^{-12} seconds.

duration(num = NULL, units = "second", ...)
一個適合自動化的持續時間 constructor. duration(5, unit = "years")

as.duration(x, ...) Coerce a timespan to a duration. Also is.duration(), is.difftime(). as.duration(i)

make_difftime(x) Make difftime with the specified number of units.
make_difftime(99999)

INTERVALS間隔

通過一個持續時間將間隔進行分割，以確定其物理長度；通過一個時間段將間隔進行分割，以確定其在鐘表時間中隱含的長度。

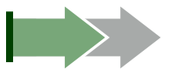
Make an interval with interval() or %--%, e.g.

Start Date | End Date

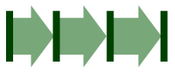
i <- interval(ymd("2017-01-01"), d) *## 2017-01-01 UTC--2017-11-28 UTC*
j <- d %--% ymd("2017-12-31") *## 2017-11-28 UTC--2017-12-31 UTC*

a %within% b 判斷間隔（interval）或日期時間（date-time）是否位於間隔（interval） b 內? now() %within% i

int_start(int)存取/設定間隔的開始日期時間。Also int_end(). int_start(i) <- now(); int_start(i)

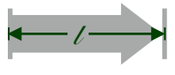


int_aligns(int1, int2) 兩個間隔是否共享一個邊界? Also int_overlaps(). int_aligns(i, j)



int_diff(times) 創建介於向量中日期時間之間的間隔。v <-c(dt, dt + 100, dt + 1000); int_diff(v)

int_flip(int)反轉一個間隔的方向。 Also int_standardize(). int_flip(i)



int_length(int) 以秒為單位表示的間隔長度 int_length(i)

int_shift(int, by) 可以將一個間隔（interval）沿著時間軸上移動一個時間跨度的長度。例如: int_shift(i, days(-1)) 可以將間隔 i 向時間軸上的前一天移動。