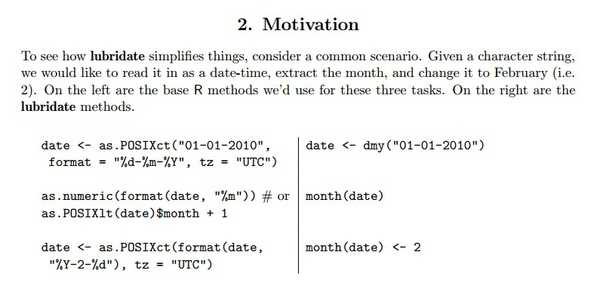
參考資料：https://zhuanlan.zhihu.com/p/27612862

**Learn R | 日期時間處理之lubridate包**

用過R的朋友們相信一定不會對**Hadley·Wickham**這個名字有所陌生，作為RStudio的Chief Scientist，他為R用戶貢獻了多個重量級的package（ggplot2、dplyr等等）。而今天介紹的這個lubridate包，也是由他所編寫，專注於對日期時間資料（Date-time data）的處理。

對於日期時間資料，R在基礎包中提供了兩種類型的時間資料，一類是Date日期資料，它不包括時間和時區資訊；另一類是POSIXct/POSIXlt類型資料，其中包括了日期、時間和時區資訊。一般來講，R語言中建立時序資料是通過字元型轉化而來，但由於時序資料形式多樣，而且R中存貯格式也是五花八門，例如Date/ts/xts/zoo/tis/fts等等。使用者很容易被一系列的資料格式所迷惑，所以時序資料的轉化和操作並不是非常方便。

為此，我們引入協力廠商包lubridate，該包主要有兩類函數，一類用於處理時點數據（time instants），另一類則用於處理時段資料（time spans）。雖然這些基礎功能R Base也能實現，但實現方式及其繁瑣，通過下圖的對比，我們可以看到同樣是時間資料處理，lubridate包比R的基礎包的操作是何等的簡潔。

一、解析日期與時間（Parsing dates and times）

首先，在使用lubridate包識別日期前，我們需要告訴它年（y）月（m）日（d）的排列順序，而函數本身就是這幾個字母的組合，非常形象：

**>** ymd(20170629);myd(06201729);dmy(29062017)

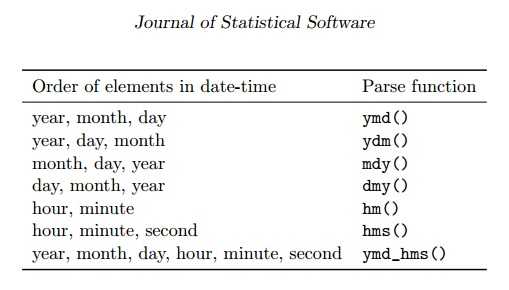
[1] "2017-06-29"

[1] "2017-06-29"

[1] "2017-06-29"

在此基礎上，如果需要讀取含具體時間的資料，就需要在函數裡再加上小時（h）分鐘（m）和秒（s）；如果需要讀入的時間具有特定時區，那就用tz選項來指定。

> test\_date <- ymd\_hms("2017-06-29-12-01-30", tz = "Pacific/Auckland")

> test\_date

[1] "2017-06-29 12:01:30 NZST"

所有公式如右表：

lubriadate非常靈活，它可以“智慧”的判斷我們的輸入格式，最好的得到標準的時間格式，甚至即使你的輸入不完全，通過一個truncated選項， 也可以識別不完整資訊的日期輸入格式。

> test\_date<- c(20170601, "2017-06-02", "2017 06 03", "2017-6-4","2017-6, 5", "Created on 2017 6 6", "201706 !!! 07")

> ymd(test\_date)

[1] "2017-06-01" "2017-06-02" "2017-06-03" "2017-06-04" "2017-06-05" "2017-06-06" "2017-06-07"

在上面的這個例子中，日期時間資料雖然雜亂，但還是按照年、月、日的順序排列的，但如果拿到的資料年月日是無序排列的呢？

這裡就需要介紹一個新的函數parse\_date\_time()，它可以將格式各樣的日期時間字元轉換為日期時間類型的資料。該函數中有一個重要的參數，即orders，通過該參數指定可能的日期格式順序，如年-月-日或月-日-年等順序，讓系統去嘗試。

> test\_date <- c('20131113','120315','12/17/1996','09-01-01','2015 12 23','2009-1, 5','Created on 2013 4 6')

> parse\_date\_time(test\_date,order = c('ymd','mdy','dmy','ymd'))

[1] "2013-11-13 UTC" "2012-03-15 UTC" "1996-12-17 UTC" "2009-01-01 UTC" "2015-12-23 UTC"

[6] "2009-01-05 UTC" "2013-04-06 UTC"

二、設置與提取資訊（Setting and Extracting information）

1. 精確提取

從日期時間提取資訊的函數也非常直觀，second()，minute()，hour()，day()，wday()，yday()，week()，month()，year()，tz()分別可以提取秒，分，小時，天，周的第幾天，年的第幾天，星期，月，年和時區的資訊。

這些函數同樣也可以用來設置修改這些資訊。其中，wday()和month()這兩個功能有一個label的選項，可以選擇顯示數值或是名字（eg：wday()可顯示7或者Sat。注：weekday默認周日為1）

> test <- ymd\_hms('2017/06/29/12/00/00')

> test

[1] "2017-06-29 12:00:00 UTC"

> second(test) <- 30

> test

[1] "2017-06-29 12:00:30 UTC"

> wday(test)

[1] 5

> wday(test,label = TRUE)

[1] Thurs

Levels: Sun < Mon < Tues < Wed < Thurs < Fri < Sat

2. 模糊提取（取整）

模糊取整即截斷函數，即將日期時間型資料取整到不同的單位，如年、季、月、日、時等。

> round\_date() # 四捨五入取整

> floor\_date() # 向下取整

> ceiling\_date() # 向上取整

> test\_date <- as.POSIXct("2017-06-29 12:34:59")

> round\_date(test\_date,'hour') [1] "2017-06-29 13:00:00 CST"

> ceiling\_date(test\_date,'hour') [1] "2017-06-29 13:00:00 CST"

> floor\_date(test\_date,'hour') [1] "2017-06-29 12:00:00 CST"

# 自訂提取

> x <- as.POSIXct("2017-06-29 12:01:59.23")

> round\_date(x, "second") [1] "2017-06-29 12:01:59 CST"

> round\_date(x, "minute") [1] "2017-06-29 12:02:00 CST"

> round\_date(x, "2 hours") [1] "2017-06-29 12:00:00 CST"

> round\_date(x, "halfyear") [1] "2017-07-01 CST"

> round\_date(x, "month") [1] "2017-07-01 CST"

> round\_date(x, "5 mins") [1] "2017-06-29 12:00:00 CST"

# 向上向下取整同理

三、時區（Time Zones）

在lubridate包中，與時區相關的function主要做兩件事。

其一，顯示同一個時間點在不同時區的時間；其二，結合某個時間點和給定時區，新建一個給定時區的時間點。簡單來講第一個是變換時區，用with\_tz()；第二個是固定時區，用force\_tz()。

> test\_date <- ymd\_hms("2017-06-29 09:00:00", tz = "Pacific/Auckland")

> with\_tz(test\_date,"America/New\_York")

[1] "2017-06-28 17:00:00 EDT"

# 給定時間和時區，新建一個給定時區的對應時間點

> test\_date <- ymd\_hms("2017-06-29 12:00:00", tz = "America/Chicago")

> test\_date #[1] "2017-06-29 12:00:00 CDT"

> test\_date\_1 <- force\_tz(test\_date,tz="Europe/London")

> test\_date\_1 #[1] "2017-06-29 12:00:00 BST"

四、時間間隔（Time Intervals）

> begin1 <- ymd\_hms("20150903,12:00:00")

> end1 <- ymd\_hms("20160804,12;30:00")

> begin2 <- ymd\_hms("20151203,12:00:00")

> end2 <- ymd\_hms("20160904,12;30:00")

> test\_date\_1 <- interval(begin1,end1)

> test\_date\_1 #[1] 2015-09-03 12:00:00 UTC--2016-08-04 12:30:00 UTC

> test\_date\_2 <- interval(begin2,end2)

# 判斷兩段時間是否有重疊

> int\_overlaps(test\_date\_1,test\_date\_2)

[1] TRUE

其他操作時間間隔的函數還包括：int\_start，int\_end，int\_flip，int\_shift，int\_aligns，union，intersect和%within%等。

在得到時間間隔的資料後，我們還可以通過time\_length()函數進一步計算間隔內的不同度量單位下的時間：

> time\_length(test\_date\_1,'day') [1] 336.0208

> time\_length(test\_date\_1,'year') [1] 0.9180897

> time\_length(test\_date\_1,'month') [1] 11.03293

> time\_length(test\_date\_1,'seconds') [1] 29032200

五、日期時間的計算（Arithmetic with date times）

1. 時間跨度（durations和periods）

時間間隔是特定的時間跨度（因為它綁定在特定時間點上）。lubridate同時也提供了一般的時間跨度的類：durations和periods。建立periods的function是用時間單位（複數）來命名的。而建立duration的function命名和periods的一致，僅在首碼加一個‘d'。

# periods

> minutes(1)

[1] "1M 0S"

# durations[加首碼'd']

> dminutes(1)

[1] "60s"

為什麼要這兩個不同的類呢？因為時間線（timeline）並沒有數字線（number line）那樣可靠。durations類通常提供了更準確的運算結果。一個duration年總是等於365天。而periods是隨著時間線的波動而給出更理性的結果，這一特點在建立時鐘時間（clock times）的模型時非常有用。比方說，durations遇到閏年時，結果就太死板，而periods給出的結果就靈活很多：

> leap\_year(2016)

[1] TRUE

> ymd(20160101)+years(1)

[1] "2017-01-01"

> ymd(20160101)+dyears(1)

[1] "2016-12-31"

我們可以用periods或者durations來做基本的日期運算。例如：得出接下來的六周的一個相同時間點：

> test\_date <- test\_date + weeks(0:5)

> test\_date

[1] "2017-06-29 12:00:00 CDT" "2017-07-06 12:00:00 CDT" "2017-07-13 12:00:00 CDT"

[4] "2017-07-20 12:00:00 CDT" "2017-07-27 12:00:00 CDT" "2017-08-03 12:00:00 CDT"

> test\_date\_2 / ddays(1) #[1] 276.0208

> test\_date\_2 / dminutes(5) #[1] 79494

# 取整

> test\_date\_2 %/% months(1) #[1] 9

# 取餘

> test\_date\_2 %% months(1)

[1] 2016-09-03 12:00:00 UTC--2016-09-04 12:30:00 UTC

用時間間隔為模數會得到一個餘數，它是一個新的時間間隔。可以用as.period()把這個時間間隔轉變為period類（非特定時間跨度的類）。

> as.period(test\_date\_2 %% months(1))

[1] "1d 0H 30M 0S"

2. %m+%

在時間計算時，由於日期資料的特殊性，如果我們需要得到每個月的最後一天的日期資料，直接在某一個月的最後一天上加上月份很明顯是錯誤的。為此我們引入%m+%函數：

> test\_date\_0 <- as.Date('2015-01-31')

> test\_date\_2 <- test\_date\_0 %m+% months(0:11)

> test\_date\_2

[1] "2015-01-31" "2015-02-28" "2015-03-31" "2015-04-30" "2015-05-31" "2015-06-30"

[7] "2015-07-31" "2015-08-31" "2015-09-30" "2015-10-31" "2015-11-30" "2015-12-31"