第 7 章: 日期函式

7: Functions for Dates

日期時間在醫學研究中, 是一個非常重要的變數, 然而對於日期時間的儲存, 列印與計算, 每一個軟體各不相同的處理方法. 原則上, 軟體對於日期時間的輸入通常是以文字型式輸入 日歷時間 (calendar date and time), 例如, "12/10/1979", "12/10/1979 20:30:10", "2/28/1947", "2-28-1947", 022847, 2Feb47, "Feburary 2 1947" 等, R 讀入文字型式的日歷時間, 轉換成 R 的 "日期時間類別物件 (date-time class object). R 的 "日期類別物件" (dates class object). R 有不同日期時間類別格式存在, 非別爲 Date, 以及 Date-Time 類別, 而 Time 包含 POSIXlt 與 POSIXct 等 2 時間類別. Date 類別 僅考慮以天數計算, Date-Time 類別考慮以總秒數計算, R 以 January 1, 1970 爲 第 0 天.

使用日期函式 as.Date() 可以將文字型式的日歷日期 (calendar date) 轉換成 R 的 日期類別 (Dates class) 物件. 使用日期函式 strptime() 可以將文字型式的日歷時間轉換成 R 的 "日期-時間類別物件"; 使用日期函式 as.POSIXlt() 與 as.POSIXct() 可以將 "日期-時間類別物件" 分別轉換成 POSIXlt, POSIXct 類別格式. 使用日期函式 format() 可以將 R 的 "日期-時間類別物件" 轉換成一般人可讀的文字型式的日期, 日, 星期, 月, 與時間. POSIXct: 非常大的正整數, 通常儲存在 data.frame 中使用, POSIXlt: 是一個列表物件 (list), 分別儲存 星期中的第幾天,

一年中的第幾天, 月份, 月中的第幾天等等. R 的日期時間的內設有些複雜, 因此 R 有些專門日期時間套件, 如 date, lubridate 等, 這些套件可以在處裡時間上變得較爲方便.

7.1 地區設定與時區設定

R 在台灣當地時間 (local time) 中文格式有時會出現 NA, 一些地區中文的介面對於如何呈現日期時間有時會出現錯誤, 若如期時間輸入輸出格式出現錯誤, 可先改正輸入輸出格式, 例如, 改用英文標準時間 Sys.setlocale("LC_TIME", "C") 或是使用 ISO 時間格式. 因爲閏年, 閏秒的調整, 時間差異有時會錯, 時區不同, 計算時間差異須小心計算時間差異時, 需先格式化日期與時間, 關閉地區時間, 使用 Sys.setlocale("LC_TIME", "C"), 改成 UTC 時區 (Universal Time, Coordinated). 或 改成 Sys.setlocale("LC_TIME", "English").

```
Sys.setlocale(category = "LC_ALL", locale = "")
```

其中引數

- category: 文字. "LC_ALL", "LC_COLLATE", "LC_CTYPE", "LC_MONETARY", "LC_NUMERIC", "LC_TIME".
- locale: 文字. 設定地區文字.

```
> ## Date
> Sys.getlocale(category = "LC_TIME")
[1] "Chinese (Traditional)_Taiwan.950"
> Sys.Date()
[1] "2017-03-08"
> ## local time error
> lct <- Sys.getlocale("LC_TIME")
> as.Date("Dec 31, 1969", format = "%b %d, %Y")
[1] NA
> Sys.setlocale("LC_TIME", "C")
[1] "C"
> as.Date("Dec 31, 1969", format = "%b %d, %Y")
[1] "1969-12-31"
```

```
> Sys.setlocale("LC_TIME", lct)
[1] "Chinese (Traditional)_Taiwan.950"
> as.Date("Dec 31, 1969", format = "%b %d, %Y")
[1] NA
```

7.2 日期時間函式: as.Date() 與 julian()

使用日期函式 as.Date() 可以將文字型式的日歷日期 (calendar date) 轉換成 R 的 日期類別物件, 日期類別物件是一個擁有 date class 數值型的向量, 僅考慮以天數計算. 在 R 中, 數值型的 date class 向量是以 January 1, 1970 爲 第 0 天, 稱爲 Julian Date, R 是以 UTC 時區 (Universal Time, Coordinated) January 1, 1970, 0 時, UTC 0 分 0 秒 爲 "0" (origin), UTC 以國際原子時爲計算基準, 修正了一些時間微小的飄移, 而 UTC 與 GMT (格林威治標準時間) 相差不多. 轉換後的標準格式是 yyyy-mm-dd, 若要查看數值型的 Julian 格式, 可用 as,numeric() 或 julian().

```
as.Date(x, ...)
## S3 method for class 'character'
as.Date(x.char, format, ...)
```

其中引數爲

- x.char: 文字向量, 顯示日期.
- format: 日期輸入格式. 詳細格式參見 表 7.2 – 表 7.3,

常用格式爲:

格式	說明
%d	日, 1-31, Day of the month (decimal number)
%m	月, 1-12, Month (decimal number)
%b	月, Jan-Dec, Month (abbreviated)
%B	月, January-December, Month (full name)
%y	年, 80, Year (2 digit)
%Y	年, 1980, Year (4 digit)

```
> ## as.Date
> x.chr <- c("1969-12-31", "1970-01-01", "1970-01-02")
> class(x.chr)
[1] "character"
> x.Date <- as.Date(x.chr)
> x.Date
[1] "1969-12-31" "1970-01-01" "1970-01-02"
> > format(x.Date, "%b %d, %Y")
[1] "Dec 31, 1969" "Jan 01, 1970"   "Jan 02, 1970"
> class(x.Date)
[1] "Date"
> str(x.Date)
 Date[1:3], format: "1969-12-31" "1970-01-01" "1970-01-02"
> #
> julian(x.Date)
[1] -1  0  1
attr(,"origin")
[1] "1970-01-01"
> unclass(x.Date)
[1] -1  0  1
> #
> unclass(x.Date)
[1] -1  0  1
>
> ## different format
> x.chr <- c("12/31/1969", "1/1/1970", "01/02/1970")
> class(x.chr)
[1] "character"
> x.Date <- as.Date(x.chr, format = "%m/%d/%Y")
> x.Date
[1] "1969-12-31" "1970-01-01" "1970-01-02"
>
> ## different format
> ## ## local time error
> x.chr <- c("Dec 31, 1969", "Jan 1, 1970", "Jan 2, 1970")
```

```
> class(x.chr)
[1] "character"
> x.Date <- as.Date(x.chr, format = "%b %d, %Y")
> x.Date # error
[1] NA NA NA
> #
> lct <- Sys.getlocale("LC_TIME")
> Sys.setlocale("LC_TIME", "C")
[1] "C"
> as.Date(x.chr, format = "%b %d, %Y") # correct
[1] "1969-12-31" "1970-01-01" "1970-01-02"
> Sys.setlocale("LC_TIME", lct)
[1] "Chinese (Traditional)_Taiwan.950"
> as.Date(x.chr, format = "%b %d, %Y") # error
[1] NA NA NA
> #
> ## difference
> as.Date("2017-03-1") - as.Date("2017-02-27")
Time difference of 2 days
```

R 是以 UTC 時區 (Universal Time, Coordinated) January 1, 1970, 0 時, UTC 0 分 0 秒 爲 “0” (origin), UTC 以國際原子時爲計算基準, 修正了一些時間微小的飄移, 而 UTC 與 GMT (格林威治標準時間) 相差不多. 若輸入天數數值向量, 可設定起始日期 (origin), 才能以數值計算相對日期.

```
## S3 method for class 'numeric'
as.Date(x.num, origin, ...)
```

其中引數爲

- `x.num`: 整數數值向量, 顯示日數.
- `origin`: 設定起始計算日.

```
> ## as.Date(x.num, origin)
> x.num <- c(-1, 0, 1)
> as.Date(x.num, origin = "1970-01-01")
[1] "1969-12-31" "1970-01-01" "1970-01-02"
> as.Date(x.num, origin = "1960-01-01")
[1] "1959-12-31" "1960-01-01" "1960-01-02"
> as.Date(x.num, origin = "1990-01-01")
[1] "1989-12-31" "1990-01-01" "1990-01-02"
```

因爲閏年, 閏秒的調整, 時間差異有時會錯, 時區不同, 計算時間差異須小心計算時間差異時, 可以先格式化日期與時間. `Date` 格式的時間, 可以用來計算天數差異.

```
## S3 method for class 'POSIXct'
as.Date(x.char, tz = "UTC", ...)
```

其中引數爲

- x.char: 文字向量, 顯示日期.
- tz: 時區標記 (Time zone effect). 參見 https://en.wikipedia.org/wiki/List_of_tz_database_time_zones.

```
> ## these time zone names are common
> as.Date(z.Date, tz = "NZ")
[1] "2017-03-08" "2017-03-09"
> as.Date(z.Date, tz = "HST") # Hawaii
[1] "2017-03-07" "2017-03-08"
> as.Date(z.Date, tz = "Asia/Taipei")
[1] "2017-03-08" "2017-03-08"
```

7.3 日期時間函式 strptime()

日期格式 Date 只能考慮日曆時間, 使用天爲計算單位, 若要同時輸入日期時間, 使用函式 strptime(), 可以同時輸入日期時間.

```
## S3 method for class 'POSIXt'
strftime(x.chr, format = "", tz = "", usetz = FALSE, ...)
strptime(x.chr, format, tz = "")
```

- x.char: 文字向量, 顯示日期.
- tz: 時區標記 (Time zone effect). 參見 https://en.wikipedia.org/wiki/List_of_tz_database_time_zones.
- usetz = TRUE: 列引時將時區列印附屬在最後位置.

```
> ## strptime()
> lct <- Sys.getlocale("LC_TIME")
> Sys.setlocale("LC_TIME", "C")
[1] "C"
```

```
> st.chr <- c("January 10, 2014 11:40",
              "December 25, 2013 09:10")
> strptime(st.chr, "%B %d, %Y %H:%M")
[1] "2014-01-10 11:40:00 CST" "2013-12-25 09:10:00 CST"
> #
> mdt.Date <- c("2/28/1947 6:30 PM",
                "2/28/1947 6:30 PM")
> strptime(mdt.Date, format = "%m/%d/%Y %I:%M %p")
[1] "1947-02-28 18:30:00 CST" "1947-02-28 18:30:00 CST"
```

7.4 日期時間格式 POSIXlt() 與 POSIXct

日期格式 Date 只能考慮日曆時間, 使用天爲計算單位, 若須要同時使用日期時間, 則須使用 R 日期時間類別物件 (date-time class object) 儲存日期時間. R 日期時間類別物件分成 2 種不同類別格式儲存, POSIXlt 與 POSIXct 儲存日期時間. POSIXlt 是以列表 (list) 型式儲存. 而 POSIXct 是以連續數值儲存 (continuous time). POSIXlt 格式表示 R 使用可讀的列表物件 (list) (lt: legible time), 而 POSIXct 格式表示 R 使用連續型時間計算的數值格式 (ct: continuous time), 計算日期時間物件相對時間 0 之間的總秒數. 可以使用日期函式 as.numeric() 將日期時間類別物件轉換成以 January 1, 1970, 0 時, 0 分 0 秒 爲 "0", 時間零點, 加以計算. 函式 as.POSIXlt() 與 as.POSIXct() 可將日期時間文字向量, 轉換成 POSIXlt 與 POSIXct 日期時間格式.

```
## S3 method for class 'character'
as.POSIXlt(x.chr, tz = "", format, ...)
## S3 method for class 'numeric'
as.POSIXlt(x.num, tz = "", origin, ...)
## S3 method for class 'POSIXlt'
as.double(x.posi, ...)
```

其中引數爲

- x.char: 文字向量, 顯示日期.
- format: 日期輸入格式. 詳細格式參見 表 7.2 – 表 7.3,

- `x.num`: 整數數值向量, 顯示日數.
- `origin`: 設定起始計算日.
- `tz`: 時區標記 (Time zone effect). 參見 https://en.wikipedia.org/wiki/List_of_tz_database_time_zones.

R 是以 UTC 時區 (Universal Time, Coordinated) January 1, 1970, 0 時, UTC 0 分 0 秒 爲 "0" (origin), `POSIXlt` 與 `POSIXct` 永遠使用 UTC 日期時間格式儲存. 若要呈現出不同地區的時區, 則使用函式 `format()` 處理. 而台灣台灣的所屬時區比 UTC 時間快 8 小時, UTC+8 或是 GMT+8. R 在台灣時區出現爲 CST (China Standard Time). `POSIXlt` 回傳 list, 包含以下向量物件參見 表 7.1.

表 7.1: `POSIXlt` 回傳 list 包含

向量	說明
sec	0-59: seconds (秒)
min	0-59: minutes (分)
hour	0-23: hours (時)
mday	1-31: day of the month (月天數)
mon	0-11: months after the first of the year. (月)
year	Years since 1900. (年)
wday	0-6 day of the week, starting on Sunday. (週天數)
yday	0-365: day of the year. (年日數)
isdst	Daylight savings time flag. (日光節約時間) Positive if in force, zero if not, negative if unknown.
'zone'	The abbreviation for the time zone. (時區簡寫)
'gmtoff'	The offset in seconds from GMT: positive values are East of the meridian. Usually NA if unknown, but 0 could mean unknown.

```
> ## as.POSIXlt() return list
> st.datetime <- Sys.time()
> str(st.datetime)
 POSIXct[1:1], format: "2017-03-11 10:31:31.219"
> is.list(st.datetime)
[1] FALSE
> st.datetime <- as.POSIXlt(st.datetime)
> str(st.datetime)
 POSIXlt[1:1], format: "2017-03-11 10:31:31.219"
> names(st.datetime)
NULL
> names(unclass(st.datetime))
 [1] "sec"    "min"    "hour"   "mday"   "mon"    "year"   "wday"   "yday"
 [9] "isdst"  "zone"   "gmtoff"
> st.datetime$mday
[1] 11
> st.datetime$wday
[1] 6
> st.datetime$yday
[1] 69
> st.datetime$sec
[1] 31.21924
> st.datetime$min
[1] 31
> st.datetime$zone
[1] "CST"
> st.datetime$gmtoff
[1] 28800
```

7.5 日期時間函式: weekdays(), months(), quarters()

使用 julian() 日期函式 weekdays(), months(), quarters() 等, 可以求取日期類別物件的訊息.

```
## S3 method for class 'POSIXt' and 'Date'
weekdays(x, abbreviate)
months(x, abbreviate)
quarters(x, abbreviate)
julian(x, ...)
julian(x, origin = as.Date("1970-01-01"), ...)
```

其中引數

- x: class 'POSIXt' 或 'Date' 物件
- abbreviate = FALSE: 是否回傳簡寫.

```
> ## weekdays(), months(), julian()
> x.chr <- c("2/28/1947", "1/1/1970", "7/15/1987")
> x.julian <- as.Date(x.chr, format = "%m/%d/%Y")
> weekdays(x.julian)
[1] "Friday"    "Thursday"  "Wednesday"
> weekdays(x.julian, abbreviate = TRUE)
[1] "Fri" "Thu" "Wed"
> months(x.julian, abbreviate = FALSE)
[1] "February" "January"  "July"
> quarters(x.julian, abbreviate = TRUE)
[1] "Q1" "Q1" "Q3"
```

7.6 日期時間函式: Sys.Date() 與 Sys.time()

使用日期時間函式 Sys.Date() 可以取得使用電腦當下的當天日期 (date), Sys.time() 可以取得電腦正在使用 R 取得使用電腦當下的絕對當天日期時間 (absolute date-time). 例如, 用來計算受試者的年齡.

```
> ## Sys.Date(), Sys.time()
> Sys.Date()
[1] "2017-03-10"
> Sys.time()
[1] "2017-03-10 21:50:36 CST"
> #
> x.systime <- Sys.time()
> x.systime
[1] "2017-03-10 21:50:36 CST"
> p <- as.POSIXlt(x.systime)
> names(unclass(p))
 [1] "sec"    "min"    "hour"   "mday"   "mon"    "year"   "wday"   "yday"   "isdst"  "zone
    "
[11] "gmtoff"
> #
> ## Sys.Date(), calculate age as of today's date
> x.date <- c("2/28/1947", "1/1/1970", "7/15/1987")
> x.julian <- as.Date(x.date, format = "%m/%d/%Y")
> date.today <- Sys.Date()
> # the display of 'days' is displaced
> x.age.day <- (date.today-x.julian)
> x.age.yr <- x.age.day/365.25
```

```
> x.age.yr
Time differences in days
[1] 70.02875 47.18686 29.65366
> # sometimes, truncate number to get "age" as an integer
> x.age.yr <- trunc(as.numeric(x.age.yr))
> x.age.yr
[1] 70 47 29
> # create a data frame
> x.df <- data.frame(Birthday = x.date,
                     Standard = x.julian,
                     Julian = as.numeric(x.julian),
                     Age = x.age.yr)
> x.df
   Birthday   Standard Julian Age
1 2/28/1947 1947-02-28  -8343  70
2  1/1/1970 1970-01-01      0  47
3 7/15/1987 1987-07-15   6404  29
```

文字型式的日歷 (calendar date) 以各種不同型式輸入, 所以有時須要用 `as.Date()` 函式中的引數 `format` 做適當之轉換, 注意, 使用中文介面, `as.Date()` 有些時候無法讀出正確日期. 須對系統做些調整. 見表 7.2 – 表 reftab:RbasicDateFormatFunction02.

```
> ## Date with input format
> as.Date("1990-1-19") # standard format
[1] "1990-01-19"
>
> ## two digits
> as.Date("9/15/14", format = "%m/%d/%y") # avoid two digits for year
[1] "2014-09-15"
> as.Date("4 25 08", format = "%m %d %y")
[1] "2008-04-25"
> as.Date("063006", format = "%m%d%y")
[1] "2006-06-30"
>
> # chinese GUI cause problems
> as.Date("August 15, 1995", format = "%B %d, %Y")
[1] NA
> as.Date("27Aug95", format = "%d%b%y") # two digits for year
[1] NA
> as.Date("27Aug1995", format = "%d%b%Y")
[1] NA
> as.Date("August 15 1995", format = "%B %d %Y")
[1] NA
> ## read in date info in format 'ddmmmyyyy'
> ## This will give NA(s) in some locales; setting the C locale
> ## as in the commented lines will overcome this on most systems.
>
> # check local setting
```

```
> ## locale-specific version of date()
> format(Sys.time(), "%a %b %d %X %Y %Z")
# return in Chinese
>
> ## change it, avoid local (chinese)
> (lct <- Sys.getlocale("LC_TIME"))
[1] "Chinese (Traditional)_Taiwan.950"
> Sys.setlocale("LC_TIME", "C")
[1] "C"
> format(Sys.time(), "%a %b %d %X %Y %Z")
[1] "Fri Mar 10 22:09:08 2017 CST"
> # read it again
> as.Date("August 15, 1995", format = "%B %d, %Y")
[1] "1995-08-15"
> as.Date("27Aug95", format = "%d%b%y") # two digits for year
[1] "1995-08-27"
> as.Date("27Aug1995", format = "%d%b%Y")
[1] "1995-08-27"
> as.Date("August 15 1995", format = "%B %d %Y")
[1] "1995-08-15"
> Sys.setlocale("LC_TIME", lct)
[1] "Chinese (Traditional)_Taiwan.950"
```

輸入日期與時間, 可以用 `strptime()` 函式, 並合併引數 `format` 做適當之轉換, 參見輔助文件: `help(strptime)`. 計算時間差異時, 需先格式化日期與時間, 台灣當地時間 (local time) 格式有時會出現 NA, 因爲閏年, 閏秒的調整, 時間差異有時會錯, 時區不同, 計算時間差異須小心改用中央標準時間 `Sys.setlocale("LC_TIME", "C")`, 使用 `POSIXlt()` 格式成相同時區, 才能正確計算時間差異.

```
> ## date and time
> ## striptime()
> # This will give NA(s) in some locales
> Sys.getlocale("LC_TIME")
[1] "Chinese (Traditional)_Taiwan.950"
> x.chr <- c("January 10, 2014 11:40", "December 25, 2013 09:10")
> x.datetime <- strptime(x.chr, "%B %d, %Y %H:%M")
> x.datetime
[1] NA NA
> #
> lct <- Sys.getlocale("LC_TIME")
> Sys.setlocale("LC_TIME", "C")
[1] "C"
> x.chr <- c("January 10, 2014 11:40", "December 25, 2013 09:10")
> x.datetime <- strptime(x.chr, "%B %d, %Y %H:%M")
> x.datetime
[1] "2014-01-10 11:40:00 CST" "2013-12-25 09:10:00 CST"
```

```
> class(x.datetime)
[1] "POSIXlt" "POSIXt"
> Sys.setlocale("LC_TIME", lct)
[1] "Chinese (Traditional)_Taiwan.950"
> x.datetime <- strptime(x.chr, "%B %d, %Y %H:%M")
> x.datetime
[1] NA NA
>
> ## read in date/time info in format 'm/d/y h:m:s'
> date.chr <- c("02/27/92", "02/27/92", "01/14/92")
> time.chr <- c("23:03:20", "22:29:56", "01:03:30")
> datetime.chr <- paste(date.chr, time.chr)
> x.datetime <- strptime(datetime.chr, "%m/%d/%y %H:%M:%S")
> x.datetime
[1] "1992-02-27 23:03:20 CST" "1992-02-27 22:29:56 CST"
[3] "1992-01-14 01:03:30 CST"
>
> ## time with fractional seconds
> z.time <- strptime("20/2/06 11:16:16.683", "%d/%m/%y %H:%M:%OS")
> z.time # prints without fractional seconds
[1] "2006-02-20 11:16:16 CST"
> op <- options(digits.secs = 3)
> z.time
[1] "2006-02-20 11:16:16.683 CST"
>
> # time difference
> ## Warning
> lct <- Sys.getlocale("LC_TIME")
> Sys.setlocale("LC_TIME", "C")
[1] "C"
> x.date <- as.Date("2012-01-01")
> y.datetime <- strptime("9 Jan 2011 11:34:21", "%d %b %Y %H:%M:%S")
> x.date-y.datetime
Error in x.date - y.datetime : non-numeric argument to binary operator
In addition: Warning message:
Incompatible methods ("-.Date", "-.POSIXt") for "-"
> # correct
> ## Time difference of 356.3 days
> x.datetime <- as.POSIXlt(x.date)
> x.datetime-y.datetime
Time difference of 356.8511 days
> #
> ## time zone
> lct <- Sys.getlocale("LC_TIME")
> Sys.setlocale("LC_TIME", "C")
[1] "C"
> x.tw <- as.Date("2012-03-01")
> y.tw <- as.Date("2012-02-28")
> x.tw-y.tw
Time difference of 2 days
```

```
> x.tw <- as.POSIXct("2012-10-25 01:00:00") # tz = CST
> y.gmt <- as.POSIXct("2012-10-25 01:00:00", tz = "GMT")
> y.gmt-x.tw
Time difference of 8 hours
```

表 7.2: 常見日期與時間之格式 (I)

格式	說明
%a	英文星期名: 簡稱 (Abbreviated weekday name)
%A	英文星期名: 全名 (Full weekday name)
%b	英文月名: 簡稱 (Abbreviated month name)
%B	英文月名: 全名 (Full month name)
%c	日期與時間 (Date and time, locale-specific)
%d	當月的第幾天 (Day of the month as decimal number, 01-31)
%H	小時, 24 時 數字 (Hours as decimal number, 00-23).
%I	小時, 12 時 數字 (Hours as decimal number, 01-12).
%j	當年的第幾天, 數字 (Day of year as decimal number, 001-366)
%m	月, 數字 (Month as decimal number, 01-12)
%M	分, 數字 (Minute as decimal number, 00-59)
%p	上午/下午 AM/PM (indicator in the locale)
	可同時與 '%I' 使用, 不可同時與 '%H' 使用
%S	秒, 數字 (Second as decimal number 00-61)
	容許閏秒 (leap seconds)
%U	當年的第幾週, 數字 (Week of the year as decimal number, 00-53)
	當年的第一個星期日爲第一週的第一天
	using the first Sunday as day 1 of week 1.
%w	當週的第幾天, 星期日 = 0 (Weekday as decimal number, 0-6, Sunday is 0).
%W	當年的第幾週, (Week of the year as decimal number 00-53)
	當年的第一個星期一爲第一週的第一天
%x	日期 (Date, locale-specific)
%X	時間 (Time, locale-specific)
%y	年, 無世紀分別 (Year without century, 00-99)
	最好不要使用
%Y	年, 有世紀分別 (Year with century)
%z	格林威治時間 (output only, offset from Greenwich)
	因此 '-0800' 爲格林威治時間西方 8 時
%Z	時間區間, 文字呈現 (output only)

表 7.3: 常見日期與時間之格式 (II)

格式	說明
%C	世紀 (Century, 00–99): 年份 / 100 的 整數部份
%D	等同 '%m/%d/%y' (Locale-specific date format)
%e	當月的第幾天, 01-31, 一位數字前會留一空白字元
%F	年-月-日, 等同於 %Y-%m-%d, ISO 8601 的時間格式
%g	The last two digits of the week-based year (see '%V').
%G	年的第幾週, 參照 '%V'
%h	等同於 %b
%k	小時, 24 時, 一位數字前會留一空白字元
%l	小時, 12 時, 一位數字前會留一空白字元
%n	新的一行 (輸入 newline)
%r	小時, 12 時, 使用 AM/PM
%R	時-分, 時間, 等同於 '%H:%M'
%t	新的一行 (輸出 newline)
%T	時-分-秒, 時間, 等同於 '%H:%M:%S'
%u	當週的第幾天, 1–7, 星期一 = 1
%V	當年的第幾週 (Week of the year as decimal number 00-53) 若當週包含一月一日, 且至少有 4 天, 則爲當年的第一週 否則, 下一週爲當年的第一週