Fechas y horarios conlubridate : :GUÍA RÁPIDA

Fecha-hora

`2017-11-28 12:00:00`
Una fecha y hora es un punto en la línea de tiempo, almacenado como el número de segundos desde 1970-01-01 00:00:00 UTC

dt <- ***as_datetime(****1511870400****)***
"2017-11-28 12:00:00 UTC"

`2017-11-28`
Una fecha es un día almacenado como el número de días transcurridos desde 1970-01-01

d <- ***as_date(****17498****)***
"2017-11-28"

`12:00:00`
Un hms es un tiempo almacenado como el número de segundos desde 00:00:00

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

ANALIZAR FECHAS-HORAS (Convertir texto o números a fechas-horas)

1. Identifique el orden de los elementos del año (y), el mes (m), el día (d), la hora (h), el minuto (m) y el segundo (s) en los datos.
2. Utilice la siguiente función, cuyo nombre replica el pedido. Cada uno acepta un argumento tz para establecer la zona horaria, e.g. 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 para trimestre. yq("2001: Q3")

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

2:01 — hms::**hms()** Además, lubridate::**hms()**, **hm()** y **ms()**, que devuelven periodos.* hms::hms(seconds = 0, minutes = 1, hours = 2)

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

now(tzone = ""**)** Hora actual en zh (predeterminado es la zh del sistema). now()

today(tzone = ""**)** Fecha actual en un zh (predeterminado es la zh del sistema). today()

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

parse_date_time() strptime más fácil. parse_date_time("09-01-01", "ymd")

OBTENCIÓN Y CONFIGURACIÓN DE COMPONENTES

Usar una función de descriptor de acceso para obtener un componente.

Asignar a una función de descriptor de acceso para cambiar un componente en su lugar.

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

2018-01-31 11:59:59 — **date(**x**)** Componente de fecha. date(dt)

2018-01-31 11:59:59 — **year(**x**)** Año. year(dt)
isoyear(x**)** El año ISO 8601.
epiyear(x**)** Año epidemiológico.

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

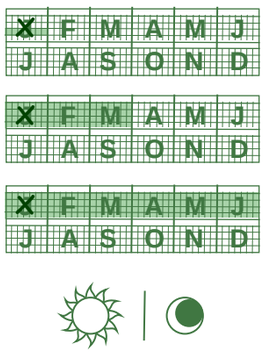
2018-01-31 11:59:59 — **day(**x**)** Día del mes. day(dt)
wday(x, label, abbr**)** Día de la semana.
qday(x**)** Día del trimestre.

2018-01-31 11:59:59 — **hour(**x**)** Hora. hour(dt)

2018-01-31 11:59:59 — **minute(**x**)** Minutos. minute(dt)

2018-01-31 11:59:59 — **second(**x**)** Segundos. second(dt)

2018-01-31 11:59:59 UTC — **tz(**x**)** Zona horaria. tz(dt)



week(x**)** Semana del año. week(dt)
isoweek() ISO 8601 semana.
epiweek() Semana epidemiológica.

quarter(x**)** Trimestre. quarter(dt)

semester(x, with_year = FALSE**)** Semestre. semester(dt)

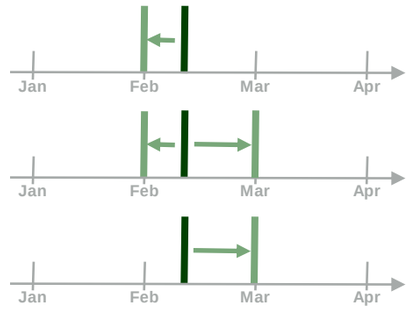
am(x**)** ¿Está en la mañana? am(dt)
pm(x**)** ¿Está en la tarde? pm(dt)

dst(x**)** ¿Es el horario de verano? dst(d)

leap_year(x**)** ¿Es un año bisiesto? leap_year(d)

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

Redondear Fechas y Horas



floor_date(x, unit = "second"**)** Redondear hacia abajo a la unidad más cercana. floor_date(dt, unit = "month")

round_date(x, unit = "second"**)** Redondear a la unidad más cercana. *round_date(dt, unit = "month")*

ceiling_date(x, unit = "second", change_on_boundary = NULL**)** Redondear hacia arriba a la unidad más cercana. ceiling_date(dt, unit = "month")

Las unidades válidas son segundo, minuto, hora, día, semana, mes, bimestre, trimestre, temporada, semestre y año.

rollback(dates, roll_to_first = FALSE, preserve_hms = TRUE**)** Retroceder al último día del mes anterior. Además, **rollforward()**. rollback(dt)

Fecha y Hora con Sello

stamp() Derive una plantilla a partir de una cadena de ejemplo y devuelva una nueva función que aplicará la plantilla a las fechas y horas. Además, **stamp_date()** y **stamp_time()**.

1. Derivar una plantilla, crear una función
   sf <- stamp("Created Sunday, Jan 17, 1999 3:34")
2. Aplicar la plantilla a las fechas
   sf(ymd("2010-04-05"))
   ## [1] "Created Monday, Apr 05, 2010 00:00"

Consejo: utilice una fecha con día > 12

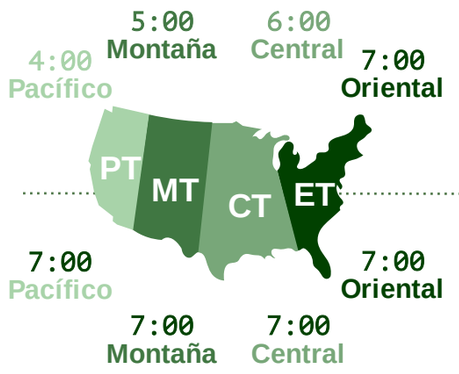
Zonas Horarias

R reconoce ~600 zonas horarias. Cada uno codifica la zona horaria, el horario de verano y las variaciones históricas del calendario para un área. R asigna una zona horaria por vector.

Utilice la zona horaria UTC para evitar el horario de verano.

OlsonNames() Devuelve una lista de nombres de zona horaria válidos. OlsonNames()

Sys.timezone() Obtiene la zona horaria actual.



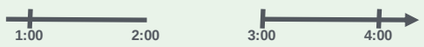
with_tz(time, tzone = ""**)** Obtenga la misma fecha y hora en una nueva zona horaria (una nueva hora de reloj). Además, **local_time(**dt, tz, units**)**. with_tz(dt, "US/Pacific")

force_tz(time, tzone = ""**)** Obtenga la misma hora del reloj en una nueva zona horaria (una nueva fecha-hora). Además, **force_tzs()**. force_tz(dt, "US/Pacific")

Matemáticas con fechas-horas — lubridate proporciona tres clases de intervalos de tiempo para facilitar las matemáticas con fechas y fechas-horas.

Las matemáticas con fechas-horas se basan en la línea de tiempo, que se comporta de manera inconsistente. Considere cómo se comporta la escala de tiempo durante:

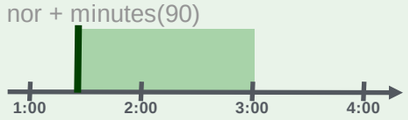
Un día normal
```
nor <- ymd_hms("2018-01-01  01:30:00",tz="US/Eastern")
```

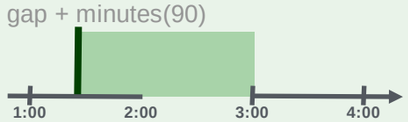

El inicio del horario de verano (primavera hacia adelante)
```
gap <- ymd_hms("2018-03-11  01:30:00",tz="US/Eastern")
```


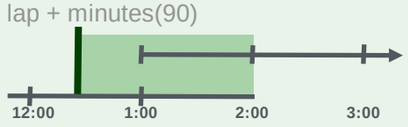
El fin del horario de verano (retroceso)
```
lap <- ymd_hms("2018-11-04  00:30:00",tz="US/Eastern")
```

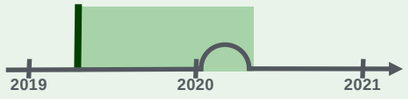

Años bisiestos y segundos bisiestos
```
leap <- ymd("2019-03-01")
```


Los períodos realizan un seguimiento de los cambios en las horas del reloj, que ignoran las irregularidades de la línea de tiempo.

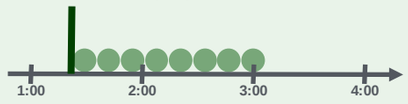
nor + minutes(90)


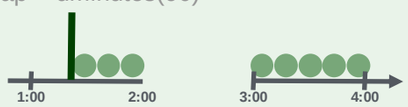
gap + minutes(90)


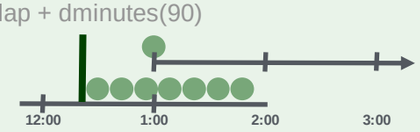
lap + minutes(90)


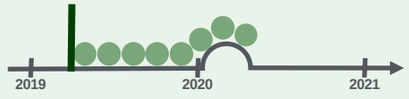
leap + years(1)


Las duraciones rastrean el paso del tiempo físico, que se desvía del tiempo del reloj cuando se producen irregularidades.

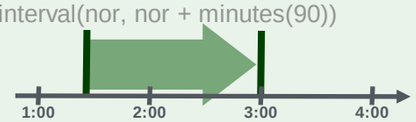
nor + dminutes(90)


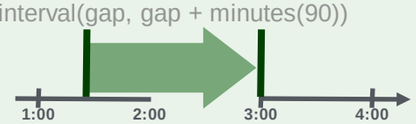
gap + dminutes(90)


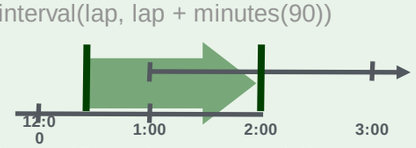
lap + dminutes(90)


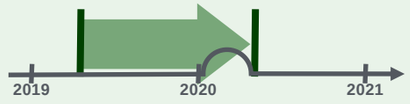
leap + dyears(1)


Los intervalos representan intervalos específicos de la escala de tiempo, delimitados por fechas y horas de inicio y finalización.

interval(nor, nor + minutes(90))


interval(gap, gap + minutes(90))


interval(lap, lap + minutes(90))


interval(leap, leap + years(1))


No todos los años son 365 días debido a los días bisiestos.

No todos los minutos son 60 segundos debido a segundos intercalares.

Es posible crear una fecha imaginaria añadiendo meses, por ejemplo, el 31 de febrero

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

%m+% and **%m-%** tirará fechas imaginarias hasta el último día del mes anterior.

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

add_with_rollback(e1, e2, roll_to_first = TRUE**)** rodará fechas imaginarias hasta el primer día del nuevo mes.

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

PERÍODOS

Agregue o reste períodos para modelar eventos que ocurren en horas específicas del reloj, como la campana de apertura de la Bolsa de Nueva York.

Haga un punto con el nombre de una unidad de tiempo pluralizado, p. ej.

```
p <- months(3) + days(12)
p
```
"3m 12d 0H 0M 0S"

Número de meses | Número de días | etc.

years(x = 1**)** x años.
months(x**)** x meses.
weeks(x = 1**)** x semanas.
days(x = 1**)** x días.
hours(x = 1**)** x horas.
minutes(x = 1**)** x minutos.
seconds(x = 1**)** x segundos.
milliseconds(x = 1**)** x milisegundos.
microseconds(x = 1**)** x microsegundos.
nanoseconds(x = 1**)** x nanosegundos.
picoseconds(x = 1**)** x picosegundos.

period(num = NULL, units = "second", ...**)** Un constructor de período compatible con la automatización. period(5, unit = "years")

as.period(x, unit**)** Forzar un intervalo de tiempo a un período, opcionalmente en las unidades especificadas. Además, **is.period()**. as.period(p)

period_to_seconds(x**)** Convierta un período al número "estándar" de segundos implícito en el período. Además, **seconds_to_period()**. period_to_seconds(p)

DURACIONES

Suma o resta duraciones para modelar procesos físicos, como la duración de la batería. Las duraciones se almacenan como segundos, la única unidad de tiempo con una longitud constante. Los difftimes son una clase de duraciones que se encuentran en la base R.

Haga una duración con el nombre de un período precedido por una d, p. ej.

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

Longitud exacta en segundos | Equivalente en unidades comunes

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

duration(num = NULL, units = "second", …**)** Un constructor de duración compatible con la automatización. duration(5, unit = "years")

as.duration(x, …**)** Forzar un intervalo de tiempo a una duración. Además, **is.duration()**, **is.difftime()**. as.duration(i)

make_difftime(x**)** Haga difftime con el número especificado de unidades. make_difftime(99999)

INTERVALOS

Divida un intervalo por una duración para determinar su longitud física, divida un intervalo por un período para determinar su longitud implícita en tiempo de reloj.

Haz un intervalo con **interval()** or **%--%**, e.j.

Fecha de inicio | Fecha final

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

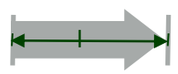

a **%within%** b ¿El intervalo o la fecha-hora a se encuentran dentro del intervalo b? now() %within% i


int_start(int**)** Acceda/establezca la fecha y hora de inicio de un intervalo. Además, **int_end()**. int_start(i) <- now(); int_start(i)


int_aligns(int1, int2**)** ¿Dos intervalos comparten un límite? Además, **int_overlaps()**. int_aligns(i, j)


int_diff(times**)** Cree los intervalos que se producen entre las fechas y horas de un vector. v <-c(dt, dt + 100, dt + 1000); int_diff(v)


int_flip(int**)** Invierte la dirección de un intervalo. Además, **int_standardize()**. int_flip(i)


int_length(int**)** Duración en segundos. int_length(i)


int_shift(int, by**)** Desplaza un intervalo hacia arriba o hacia abajo en la escala de tiempo en un intervalo de tiempo. int_shift(i, days(-1))

as.interval(x, start, …**)** Forzar un intervalo de tiempo a un intervalo con la fecha y hora de inicio. Además **is.interval()**. as.interval(days(1), start = now())