Data e hora com lubridate : : **Folha de Resumo**

Data e hora

`2017-11-28 12:00:00`
Uma data e hora é um ponto na linha do tempo gravada como o número de segundos desde 01-01-1970 00:00:00 UTC

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

`2017-11-28`
Uma data é um dia gravado como um número de dias desde 01-01-1970

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

`12:00:00`
Um hms é uma hora gravada como o número de segundos desde 00:00:00

t <- hms::as.hms(85)
00:01:25

GERAR DATA E HORA (Converte strings ou números em data e hora)

1. Identifica a ordem dos elementos de ano(y), mês(m), dia(d), hora(h), min(m), segundos(s) de sua data e hora.
2. As funções abaixo identificam esta ordem. Cada uma delas também aceita o argumento tz para o fuso-horário, ex. 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 trimestre. yq("2001: Q3")

07-2020 — my(), ym(). my("07-2020")

2:01 — hms::hms() Ver lubridate::hms(), hm() e ms(), que retornam períodos.* hms::hms(sec = 0, min= 1, hours = 2, roll = FALSE)

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

now(tzone = "") Horário atual. Por padrão no fuso do sistema. now()

today(tzone = "") Data atual. Por padrão no fuso do sistema. today()

fast_strptime() . Função strptime mas rápida. fast_strptime('9/1/01', '%y/%m/%d')

parse_date_time(). strptime simplificada. parse_date_time("9/1/01", "ymd")

OBTER E DEFINIR COMPONENTES

Use um função de acesso para obter os componentes de um data e hora.
Atribua um valor a uma função de acesso para mudar um componente.

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

2018-01-31 11:59:59 — date(x) Componente Data. date(dt)
year(x) Ano. year(dt)
isoyear(x) Ano ISO 8601.

2018-01-31 11:59:59 — epiyear(x) Ano Epidemiológico.

2018-01-31 11:59:59 — month(x, label, abbr) Mês. month(dt)
day(x) Dia do mês. day(dt)
wday(x, label, abbr) Dia da semana.

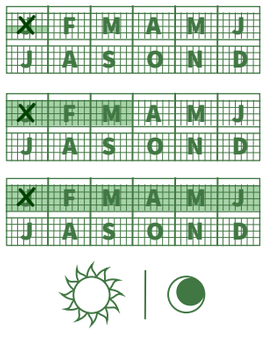
2018-01-31 11:59:59 — qday(x) Dia do trimestre.

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) Fuso-horário. tz(dt)
week(x) Semana do anor. week(dt)
isoweek() Semana ano ISO 8601.



epiweek() Semana ano Epidemiológico.

quarter(x) Trimestre. quarter(dt)

semester(x, with_year = FALSE) Semestre. semester(dt)
am(x) É manhã (am) am(dt)

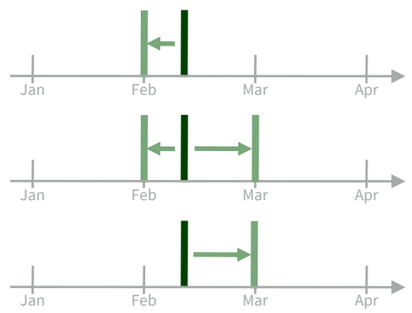
pm(x) É tarde (pm)? pm(dt)

dst(x) É horário-verão? dst(d)

leap_year(x) É ano bi-sexto? leap_year(d)

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

Arredondar Data e hora



floor_date(x, unit = "second") Arredonda para menor mais próxima. floor_date(dt, unit = "month")

round_date(x, unit = "second") Arredonda para a unidade mais próxima. round_date(dt, unit = "month")

ceiling_date(x, unit = "second", change_on_boundary = NULL) Arredondo para maior unidade mais próxima. ceiling_date(dt, unit = "month")

Unidades válidas são second, minute, hour, day, week, month, bimonth, quarter, season, halfyear e year.

rollback(dates, roll_to_first = FALSE, preserve_hms = TRUE) Retorna para o últmo dia do mês anterior. Ver rollforward(). rollback(dt)

Carimbar Data e hora

stamp() Deriva um modelo de um string de exemplo e returna uma nova função que aplica este modelo em data e hora. Ver também stamp_date() e stamp_time().

1. Deriva um modelo, cria uma função

```
sf <- stamp("Created Sunday, Jan 17, 1999 3:34")
```

2. Aplica o modelo para data e hora

```
sf(ymd("2010-04-05"))
## [1] "Created Monday, Apr 05, 2010 00:00"
```

Dica: **use uma date com dia > 12**

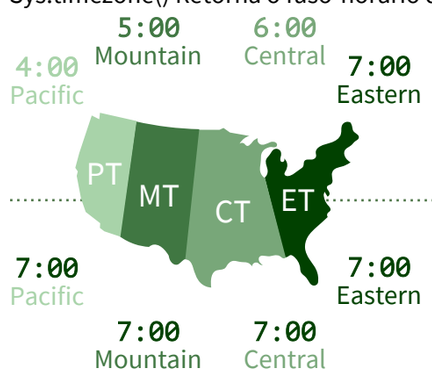
Fuso-horários

R reconhece ~600 fuso-horários. Cada um tem o fuso-horário, horários de verão e histórico de variações dos calendários de cada região. R assinala *um* fuso-horário por vetor.

Use o fuso horário UTC para evitar horário de verão.

OlsonNames() Returna uma lista dos fuso-horários válidos. OlsonNames()

Sys.timezone() Retorna o fuso-horário do sistema



with_tz(time, tzone = "") Retorna a mesma data e hora em um novo fuso-horário (um novo horário de relógio). Ver local_time(dt, tz, units). with_tz(dt, "US/Pacific")

force_tz(time, tzone = "") Retorna o mesmo horário em um novo fuso-horário (um novo data e hora). Ver force_tzs(). force_tz(dt, "US/Pacific")

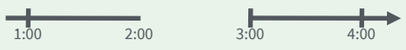
Aritmética com Data e hora

Lubridate fornece três classes de deslocamento do tempo para facilitar aritmética de datas e horários.

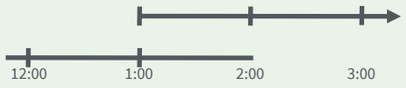
Aritmética com data e hora se baseia na linha do tempo, o que se comparta de maneira inconsistente. Considere como a linha do tempo se comporta durante:

Um dia normal
nor <- ymd_hms("2018-01-01 01:30:00",tz="US/Eastern")

No início do horário de verão
gap <- ymd_hms("2018-03-11 01:30:00",tz="US/Eastern")



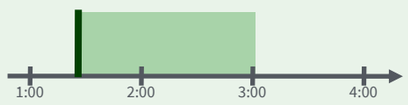
Ao final do horário de verão
lap <- ymd_hms("2018-11-04 00:30:00",tz="US/Eastern")



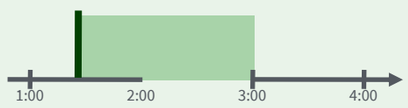
Ano bi-sexto ou segundo de ajuste
leap <- ymd("2019-03-01")

Períodos rastreiam mudanças no horário do relógio, o que ignora irregularidades da linha do tempo.

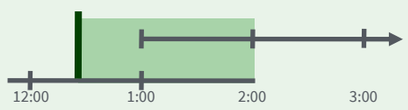
nor + minutes(90)



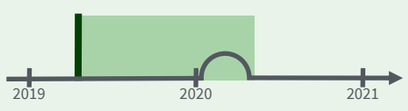
gap + minutes(90)



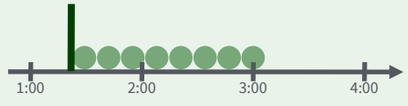
lap + minutes(90)



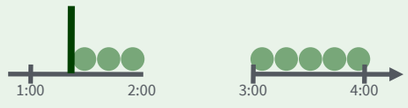
leap + years(1)



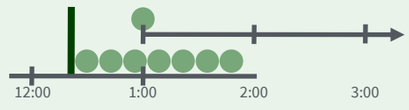
Durações rastreiam a passagem do tempo físico, o que desviam do horário do relógio quando irregularidades ocorrem.
nor + dminutes(90)



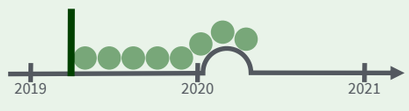
gap + dminutes(90)



lap + dminutes(90)

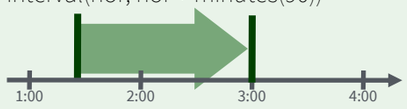


leap + dyears(1)

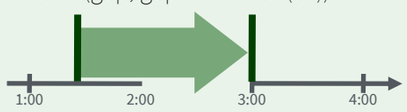


Intervalos representam um período na linha do tempo, com data e horário de início e fim.

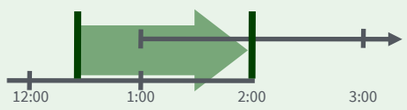
interval(nor, nor + minutes(90))



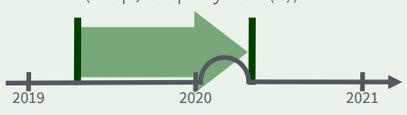
interval(gap, gap + minutes(90))



interval(lap, lap + minutes(90))



interval(leap, leap + years(1))



Nem todos os anos tem 365 dias (bissexto). Nem todos os minutos tem 60 segundos (seg. ajuste)

É possível criar data imaginárias adicionando meses a uma data, ex:. 31 de Fevereiro
jan31 <- ymd(20180131)
jan31 + months(1)

NA

%m+% e %m-% rolam as data imaginárias para o último dia do mês anteior.
jan31 %m+% months(1)

"2018-02-28"

add_with_rollback(e1, e2, roll_to_first = TRUE) rola a data imaginária para o primeiro dia do mês seguinte.
add_with_rollback(jan31, months(1), roll_to_first = TRUE)

"2018-03-01"

PERÍODOS

Adicionam ou subtraem a eventos que ocorreram em um horário específico, como o sino da abertura da bolsa (NYSE).

Cria um período com o nome da unidade de tempo no plural, ex:

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

Numero de meses | Numero de dias | etc.

years(x = 1) x anos.
months(x) x meses.
weeks(x = 1) x semanas.
days(x = 1) x dias.
hours(x = 1) x horas.
minutes(x = 1) x minutos.
seconds(x = 1) x segundos.
milliseconds(x = 1) x millisegundos.
microseconds(x = 1) x microssegundos.
nanoseconds(x = 1) x nanosegundos.
picoseconds(x = 1) x picosegundos.

period(num = NULL, units = "second", ...) Um construtor automatizado amigável.
period(5, unit = "years")

as.period(x, unit) Converte um espaço de tempo em um período, opcionalmente com unidade específica. Ver is.period().
as.period(i)

period_to_seconds(x) Converte um período para o "número de segundos padrão" presentes no período. Ver seconds_to_period().
period_to_seconds(p)

DURAÇÕES

Adicionam ou subtraem durações ao modelo físico processado, como a vida útil de uma bateria. Durações são gravadas como segundos, a única unidade consistente. Difftimes são classes do R básico.

Cria uma duração com o nome de um período precedida por um *d*, ex:

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

Tempo exato em segundos | Equivalente em unidade comum

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 seconds.
dmilliseconds(x = 1) x $\times 10^{-3}$ segundos.
dmicroseconds(x = 1) x $\times 10^{-6}$ segundos.
dnanoseconds(x = 1) x $\times 10^{-9}$ segundos.
dpicoseconds(x = 1) x $\times 10^{-12}$ segundos.

duration(num = NULL, units = "second", ...) Um construtor automatizado amigável.
duration(5, unit = "years")

as.duration(x, ...) Converte um espaço no tempo em uma duração. Ver is.duration(), is.difftime(). as.duration(i)

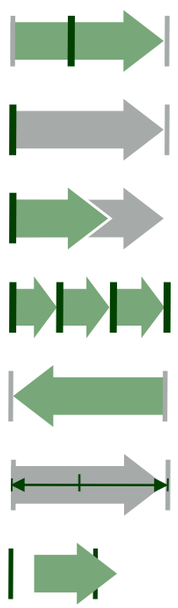
make_difftime(x) Cria um difftime com número específico de unidades.
make_difftime(99999)

INTERVALOS

Divida um interval pela duração para determinar o tempo físico, divida um intervalo por um período para determinar o tempo relativo ao horário do relógio.

Cria um intervalo com interval() ou %--%, ex:
i <- interval(ymd("2017-01-01"), d) *## 2017-01-01 UTC--2017-01-02 UTC*
j <- d %--% ymd("2017-12-31") *## 2017-11-28 UTC--2017-12-31 UTC*

Data Início | Data Fim



a %within% b O intervalo ou data e hora *a* está dentro do intervalo *b*? now() %within% i

int_start(int) Obtem/Define a data e hora do início do intervalo. Ver int_end(). int_start(i) <- now(); int_start(i)

int_aligns(int1, int2) Este dois intervalos tem o mesmo limite? Ver int_overlaps(). int_aligns(i, j)

int_diff(times) Cria um intervalo que existe em um vetor de data e hora.
v <-c(dt, dt + 100, dt + 1000); int_diff(v)

int_flip(int) Inverte a direção de um intervalo. Ver int_standardize(). int_flip(i)

int_length(int) Duração em segundos. int_length(i)

int_shift(int, by) Desloca um intervalo para frente ou para traz. int_shift(i, days(-1))

as.interval(x, start, ...) Converte um espaço no tempo em um intervalo à partir de um data de início. Also is.interval(). as.interval(days(1), start = now())