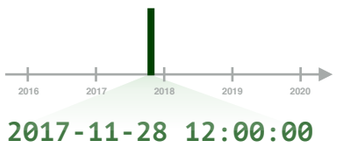
Дата и время с lubridate : : ШПАРГАЛКА

Дата-время



`2017-11-28 12:00:00`
Дата-время (**date-time**) - это точка на оси времени, хранящаяся как кол-во секунд с момента 1970-01-01 00:00:00 UTC

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

`2017-11-28`
Дата (**date**) - это день, хранящийся как кол-во дней с 1970-01-01

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

`12:00:00`
Объект hms - это **время**, хранящееся как кол-во секунд с 00:00:00

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

РАЗБОР ДАТЫ-ВРЕМЕНИ

(Конвертирование строк и чисел в date-time)

1. Определите в данных порядок следующих элементов: год (**y**), месяц (**m**), день (**d**), час (**h**), минута (**m**) и секунда (**s**).
2. Используйте одну из функций, чье имя соответствует порядку. Каждая принимает мн-во различных форматов.

`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 - квартал. *yq("2001: Q3")*

`2:01` hms::**hms**() Также lubridate::**hms**(), **hm**() и **ms**(), которые возвращают периоды.* *hms::hms(sec = 0, min= 1, hours = 2)*

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

now(tzone = "") Текущее время в час. поясе (по умол. - системный). *now()*

today(tzone = "") Текущая дата в час. поясе (по умол. - системный). *today()*

fast_strptime() Быстрый strptime. *fast_strptime('9/1/01', '%y/%m/%d')*

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

ПОЛУЧЕНИЕ И ЗАДАНИЕ КОМПОНЕНТОВ

Используйте функции доступа для получения компонента.

d ## "2017-11-28"
day(d) ## 28

Присваивайте результату функции доступа для изменения компонента на месте.

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

`2018-01-31 11:59:59` **date**(x) Дата. *date(dt)*

`2018-01-31 11:59:59` **year**(x) Год. *year(dt)*
isoyear(x) Год в ISO 8601.
epiyear(x) Эпидемиол. год.

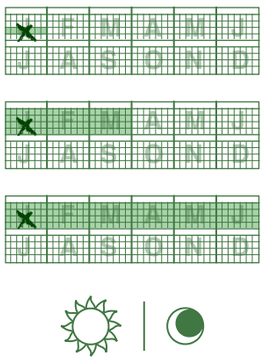
`2018-01-31 11:59:59` **month**(x, label, abbr) Месяц. *month(dt)*

`2018-01-31 11:59:59` **day**(x) День месяца. *day(dt)*
wday(x,label,abbr) День недели.
qday(x) День квартала.

`2018-01-31 11:59:59` **hour**(x) Часы. *hour(dt)*

`2018-01-31 11:59:59` **minute**(x) Минуты. *minute(dt)*

`2018-01-31 11:59:59` **second**(x) Секунды. *second(dt)*



week(x) Неделя года. *week(dt)*
isoweek() Неделя ISO 8601.
epiweek() Эпидемиол. неделя.

quarter(x, with_year = FALSE) Квартал. *quarter(dt)*

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

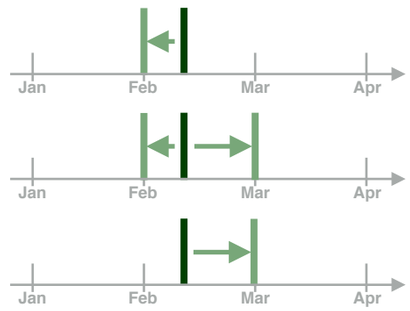
am(x) Первая пол. дня? *am(dt)*
pm(x) Вторая пол. дня? *pm(dt)*

dst(x) Летнее время? *dst(d)*

leap_year(x) Високосный год? *leap_year(d)*

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

Округления



floor_date(x, unit = "second") Округл. вниз к ближ. эл-ту. *floor_date(dt, unit = "month")*

round_date(x, unit = "second") Округ. к ближайшему эл-ту. *round_date(dt, unit = "month")*

ceiling_date(x, unit = "second", change_on_boundary = NULL) Округ. вверх к ближ. эл-ту. *ceiling_date(dt, unit = "month")*

rollback(dates, roll_to_first = FALSE, preserve_hms = TRUE) Откат к последнему дню пред. месяца. *rollback(dt)*

Шаблоны

stamp() Определяет шаблон из эталонной строки и возвращает новую функцию, которая применяет шаблон к date-time. Также **stamp_date**() и **stamp_time**().

1. Определение шаблона, создание функции
   sf <- stamp("Created Sunday, Jan 17, 1999 3:34")
2. Применение шаблона к датам
   sf(ymd("2010-04-05"))
   ## [1] "Created Monday, Apr 05, 2010 00:00"

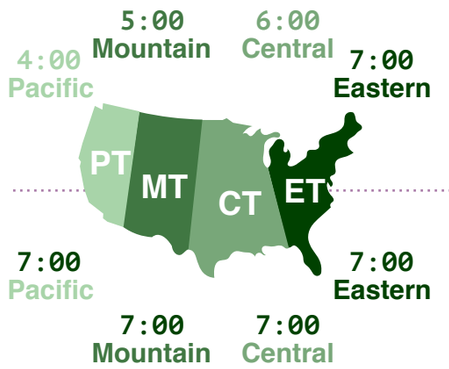
Совет: Исп. дату с day > 12

Часовые пояса

R распознает ~600 часовых поясов. Они кодируют часовой пояс, использование летнего времени и особенности календаря для области. В R используется *один* часовой пояс на вектор.

Используйте **UTC** для работы без летнего времени.

OlsonNames() Возвращает список доступных часовых поясов. *OlsonNames()*



with_tz(time, tzone = "") Возвр. **такой же date-time** в новом час. поясе (новое время "на часах"). *with_tz(dt, "US/Pacific")*

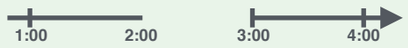
force_tz(time, tzone = "") Возвр. **такое же время "на часах"** в новом часовом поясе (новый date-time). *force_tz(dt, "US/Pacific")*

Операции с датой-временем — В lubridate есть три класса для промежутка времени для работы с датами и date-time

Операции с date-time опираются на **ось времени**, которая обладает непостоянным поведением. Примеры:

Нормальный день
nor <- ymd_hms("2018-01-01 01:30:00",tz = "US/Eastern")

Начало летнего времени (час вперед)
gap <- ymd_hms("2018-03-11 01:30:00",tz = "US/Eastern")

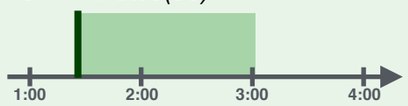


Конец летнего времени (час назад)
lap <- ymd_hms("2018-11-04 00:30:00",tz = "US/Eastern")

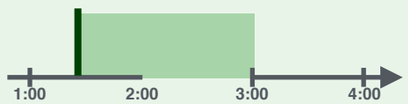
Високосные годы и секунды
leap <- ymd("2019-03-01")

Периоды отражают изменения во времени "на часах", игнорируя особенности оси времени.

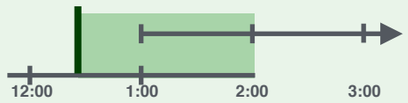
nor + minutes(90)



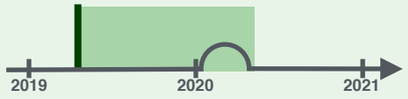
gap + minutes(90)



lap + minutes(90)

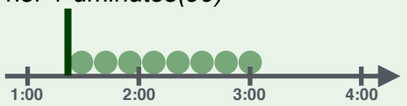


leap + years(1)



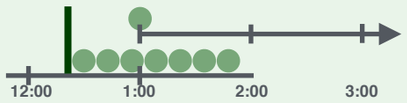
Длительности отражают прошествие физического времени, отличного от времени "на часах" в моменты особ-тей оси времени.

nor + dminutes(90)

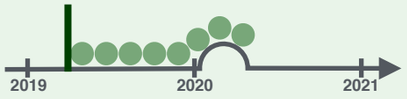


gap + dminutes(90)

lap + dminutes(90)

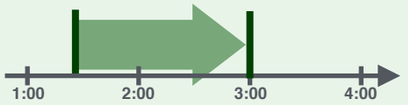


leap + dyears(1)



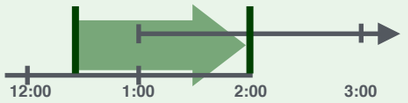
Интервалы представляют опр. интервалы оси времени, огранич. начал. и конеч. датой-временем.

interval(nor, nor + minutes(90))

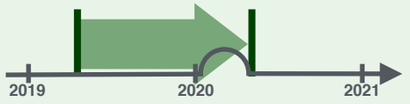


interval(gap, gap + minutes(90))

interval(lap, lap + minutes(90))



interval(leap, leap + years(1))



Не во всех годах 365 дней из-за **високос. дней**.

Не во всех минутах 60 секунд из-за **високосных секунд**.

Возможно создать нереальную дату сложением месяцев, напр. 31-е февраля

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

%m+% и **%m-%** откатывают нереальные даты к крайнему дню предшествующего месяца.

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

add_with_rollback(e1, e2, roll_to_first = TRUE) откатывает нереальные даты к первому дню нового месяца.

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

ПЕРИОДЫ

Добавляйте или вычитайте периоды для моделирования событий в определенное время "на часах", таких как открывающий звонок NYSE.

Создавайте период при помощи функции с именем **множественного числа** единицы измерения, например

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

Кол-во месяцев | Кол-во дней | и т.д.

years(x = 1) x лет.
months(x) x месяцев.
weeks(x = 1) x недель.
days(x = 1) x дней.
hours(x = 1) x часов.
minutes(x = 1) x минут.
seconds(x = 1) x секунд.
milliseconds(x = 1) x миллисекунд.
microseconds(x = 1) x микросекунд.
nanoseconds(x = 1) x наносекунд.
picoseconds(x = 1) x пикосекунд.

period(num = NULL, units = "second", ...) Конструктор периодов, подходящий для автоматизации.
period(5, unit = "years")

as.period(x, unit) Приводит промежуток времени к периоду с возможным заданием единицы измерения. Также **is.period**(). *as.period(i)*

period_to_seconds(x) Конвертирует период в "стандартное" кол-во секунд, заданных периодом. Также **seconds_to_period**().
period_to_seconds(p)

ДЛИТЕЛЬНОСТИ

Добавляйте или вычитайте длительности для моделирования физических процессов, таких как время работы батареи. Длительности хранятся в секундах, единственной единице времени с постоянной длиной. **Difftime** - класс длительности в базовом R.

Создавайте длительность при помощи функции с именем периода и префиксом ***d***, напр.

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

Точная длина в сек. | Эквивалент в обычных единицах

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

duration(num = NULL, units = "second", ...) Конструктор длительностей, подходящий для автоматизации.
duration(5, unit = "years")

as.duration(x, …) Приводит промежуток времени к длительности. Также **is.duration**(), **is.difftime**(). *as.duration(i)*

make_difftime(x) Создает difftime с заданным количеством единиц.
make_difftime(99999)

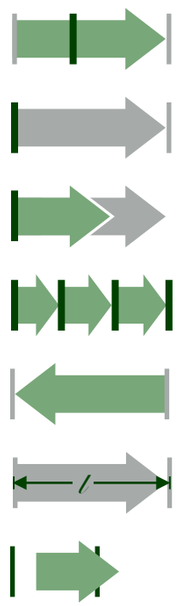
ИНТЕРВАЛЫ

Делите интервал на длительность для вычисления его физической продолжительности, а на период - для вычисления предполагаемой продолжительности "на часах".

Создавайте интервал с **interval**() или **%--%**, напр.

Нач. дата | Кон. дата

*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 Попадает ли дата-время *a* в интервал *b*? *now() %within% i*

int_start(int) Получение/задание начальной даты-времени интервала. Также **int_end**(). *int_start(i) <- now(); int_start(i)*

int_aligns(int1, int2) Обладают ли два интервала общей границей? Также **int_overlaps**(). *int_aligns(i, j)*

int_diff(times) Создает интервалы между эл-тами date-time вектора.
v <-c(dt, dt + 100, dt + 1000); int_diff(v)

int_flip(int) Изменяет направление интервала. Также **int_standardize**(). *int_flip(i)*

int_length(int) Продолжительность в секундах. *int_length(i)*

int_shift(int, by) Смещает интервал вперед или назад по оси времени на промежуток. *int_shift(i, days(-1))*

as.interval(x, start, …) Приводит промежуток времени к интервалу с начальной датой-временем. Также **is.interval**().
as.interval(days(1), start = now())