

## Dates and Times

## Packages for this section

```
library(tidyverse)  
# library(lubridate)
```

lubridate is the package that handles dates and times, but is now part of the tidyverse, so no need to load separately.

## Dates

- ▶ Dates represented on computers as “days since an origin”, typically Jan 1, 1970, with a negative date being before the origin:

```
mydates <- c("1970-01-01", "2007-09-04", "1931-08-05")
(somedates <- tibble(text = mydates) %>%
  mutate(
    d = as.Date(text),
    numbers = as.numeric(d)
  ))
```

```
# A tibble: 3 x 3
```

|   | text       | d          | numbers |
|---|------------|------------|---------|
|   | <chr>      | <date>     | <dbl>   |
| 1 | 1970-01-01 | 1970-01-01 | 0       |
| 2 | 2007-09-04 | 2007-09-04 | 13760   |
| 3 | 1931-08-05 | 1931-08-05 | -14029  |

## Doing arithmetic with dates

- ▶ Dates are “actually” numbers, so can add and subtract (difference is 2007 date in d minus others):

```
somedates %>% mutate(plus30 = d + 30, diffs = d[2] - d)
```

```
# A tibble: 3 x 5
```

|   | text       | d          | numbers | plus30     | diffs      |
|---|------------|------------|---------|------------|------------|
|   | <chr>      | <date>     | <dbl>   | <date>     | <drtn>     |
| 1 | 1970-01-01 | 1970-01-01 | 0       | 1970-01-31 | 13760 days |
| 2 | 2007-09-04 | 2007-09-04 | 13760   | 2007-10-04 | 0 days     |
| 3 | 1931-08-05 | 1931-08-05 | -14029  | 1931-09-04 | 27789 days |

## Reading in dates from a file

- ▶ `read_csv` and the others can guess that you have dates, if you format them as year-month-day, like column 1 of this `.csv`:

```
date,status,dunno
2011-08-03,hello,August 3 2011
2011-11-15,still here,November 15 2011
2012-02-01,goodbye,February 1 2012
```

- ▶ Then read them in:

```
my_url <- "http://ritsokiguess.site/datafiles/mydates.csv"
ddd <- read_csv(my_url)
```

- ▶ `read_csv` guessed that the 1st column is dates, but not 3rd.

## The data as read in

```
ddd
```

```
# A tibble: 3 x 3
```

```
  date      status      dunno
```

```
  <date>    <chr>      <chr>
```

```
1 2011-08-03 hello      August 3 2011
```

```
2 2011-11-15 still here November 15 2011
```

```
3 2012-02-01 goodbye    February 1 2012
```

## Dates in other formats

- ▶ Preceding shows that dates should be stored as text in format yyyy-mm-dd (ISO standard).
- ▶ To deal with dates in other formats, use package lubridate and convert. For example, dates in US format with month first:

```
tibble(usdates = c("05/27/2012", "01/03/2016", "12/31/2015"),  
       mutate(iso = mdy(usdates)))
```

```
# A tibble: 3 x 2  
  usdates      iso  
  <chr>       <date>  
1 05/27/2012 2012-05-27  
2 01/03/2016 2016-01-03  
3 12/31/2015 2015-12-31
```

## Trying to read these as UK dates

```
tibble(usdates = c("05/27/2012", "01/03/2016", "12/31/2015"),  
       mutate(uk = dmy(usdates)))
```

```
# A tibble: 3 x 2  
  usdates      uk  
  <chr>        <date>  
1 05/27/2012 NA  
2 01/03/2016 2016-03-01  
3 12/31/2015 NA
```

- ▶ For UK-format dates with month second, one of these dates is legit, but the other two make no sense.



## Our data frame's last column:

► Back to this:

```
ddd
```

```
# A tibble: 3 x 3
```

```
  date      status      dunno  
  <date>    <chr>      <chr>
```

```
1 2011-08-03 hello      August 3 2011
```

```
2 2011-11-15 still here November 15 2011
```

```
3 2012-02-01 goodbye    February 1 2012
```

► Month, day, year in that order.

so interpret as such

```
(ddd %>% mutate(date2 = mdy(dunno)) -> d4)
```

```
# A tibble: 3 x 4
```

|   | date       | status     | dunno            | date2      |
|---|------------|------------|------------------|------------|
|   | <date>     | <chr>      | <chr>            | <date>     |
| 1 | 2011-08-03 | hello      | August 3 2011    | 2011-08-03 |
| 2 | 2011-11-15 | still here | November 15 2011 | 2011-11-15 |
| 3 | 2012-02-01 | goodbye    | February 1 2012  | 2012-02-01 |

## Are they really the same?

- ▶ Column date2 was correctly converted from column dunno:

```
d4 %>% mutate(equal = identical(date, date2))
```

```
# A tibble: 3 x 5
```

|   | date       | status     | dunno            | date2      | equal |
|---|------------|------------|------------------|------------|-------|
|   | <date>     | <chr>      | <chr>            | <date>     | <lgl> |
| 1 | 2011-08-03 | hello      | August 3 2011    | 2011-08-03 | TRUE  |
| 2 | 2011-11-15 | still here | November 15 2011 | 2011-11-15 | TRUE  |
| 3 | 2012-02-01 | goodbye    | February 1 2012  | 2012-02-01 | TRUE  |

- ▶ The two columns of dates are all the same.

## Making dates from pieces

Starting from this file:

```
year month day
```

```
1970 1 1
```

```
2007 9 4
```

```
1940 4 15
```

```
my_url <- "http://ritsokiguess.site/datafiles/pieces.txt"  
dates0 <- read_delim(my_url, " ")
```

## Making some dates

```
dates0
```

```
# A tibble: 3 x 3
  year month  day
<dbl> <dbl> <dbl>
1  1970     1     1
2  2007     9     4
3  1940     4    15
```

```
dates0 %>%
  unite(dates, day, month, year)%>%
  mutate(d = dmy(dates)) -> newdates
```

## The results

```
newdates
```

```
# A tibble: 3 x 2
  dates      d
  <chr>     <date>
1 1_1_1970 1970-01-01
2 4_9_2007 2007-09-04
3 15_4_1940 1940-04-15
```

- ▶ `unite` glues things together with an underscore between them (if you don't specify anything else). Syntax: first thing is new column to be created, other columns are what to make it out of.
- ▶ `unite` makes the original variable columns year, month, day *disappear*.
- ▶ The column `dates` is text, while `d` is a real date.

## Extracting information from dates

```
newdates %>%  
  mutate(  
    mon = month(d),  
    day = day(d),  
    weekday = wday(d, label = TRUE)  
  )
```

# A tibble: 3 x 5

|   | dates     | d          | mon   | day   | weekday |
|---|-----------|------------|-------|-------|---------|
|   | <chr>     | <date>     | <dbl> | <int> | <ord>   |
| 1 | 1_1_1970  | 1970-01-01 | 1     | 1     | Thu     |
| 2 | 4_9_2007  | 2007-09-04 | 9     | 4     | Tue     |
| 3 | 15_4_1940 | 1940-04-15 | 4     | 15    | Mon     |

## Dates and times

- ▶ Standard format for times is to put the time after the date, hours, minutes, seconds:

```
(dd <- tibble(text = c(
  "1970-01-01 07:50:01", "2007-09-04 15:30:00",
  "1940-04-15 06:45:10", "2016-02-10 12:26:40"
)))
```

```
# A tibble: 4 x 1
```

```
text
```

```
<chr>
```

```
1 1970-01-01 07:50:01
```

```
2 2007-09-04 15:30:00
```

```
3 1940-04-15 06:45:10
```

```
4 2016-02-10 12:26:40
```



## Converting text to date-times:

- ▶ Then get from this text using `ymd_hms`:

```
dd %>% mutate(dt = ymd_hms(text))
```

```
# A tibble: 4 x 2
```

|   | text                |                     | dt    |
|---|---------------------|---------------------|-------|
|   | <chr>               |                     | <dtm> |
| 1 | 1970-01-01 07:50:01 | 1970-01-01 07:50:01 |       |
| 2 | 2007-09-04 15:30:00 | 2007-09-04 15:30:00 |       |
| 3 | 1940-04-15 06:45:10 | 1940-04-15 06:45:10 |       |
| 4 | 2016-02-10 12:26:40 | 2016-02-10 12:26:40 |       |

# Timezones

- ▶ Default timezone is “Universal Coordinated Time”. Change it via `tz=` and the name of a timezone:

```
dd %>%  
  mutate(dt = ymd_hms(text, tz = "America/Toronto")) -> dd  
dd %>% mutate(zone = tz(dt))
```

```
# A tibble: 4 x 3
```

|   | text                |  | dt                  |  | zone            |
|---|---------------------|--|---------------------|--|-----------------|
|   | <chr>               |  | <dtm>               |  | <chr>           |
| 1 | 1970-01-01 07:50:01 |  | 1970-01-01 07:50:01 |  | America/Toronto |
| 2 | 2007-09-04 15:30:00 |  | 2007-09-04 15:30:00 |  | America/Toronto |
| 3 | 1940-04-15 06:45:10 |  | 1940-04-15 06:45:10 |  | America/Toronto |
| 4 | 2016-02-10 12:26:40 |  | 2016-02-10 12:26:40 |  | America/Toronto |

## Extracting time parts

► As you would expect:

```
dd %>%
  select(-text) %>%
  mutate(
    h = hour(dt),
    sec = second(dt),
    min = minute(dt),
    zone = tz(dt)
  )
```

# A tibble: 4 x 5

|   | dt                  | h     | sec   | min   | zone            |
|---|---------------------|-------|-------|-------|-----------------|
|   | <dtm>               | <int> | <dbl> | <int> | <chr>           |
| 1 | 1970-01-01 07:50:01 | 7     | 1     | 50    | America/Toronto |
| 2 | 2007-09-04 15:30:00 | 15    | 0     | 30    | America/Toronto |
| 3 | 1940-04-15 06:45:10 | 6     | 10    | 45    | America/Toronto |
| 4 | 2016-02-10 12:26:40 | 12    | 40    | 26    | America/Toronto |

## Same times, but different time zone:

```
dd %>%  
  select(dt) %>%  
  mutate(oz = with_tz(dt, "Australia/Sydney"))
```

# A tibble: 4 x 2

|   | dt                  |            | oz       |
|---|---------------------|------------|----------|
|   | <dtm>               |            | <dtm>    |
| 1 | 1970-01-01 07:50:01 | 1970-01-01 | 22:50:01 |
| 2 | 2007-09-04 15:30:00 | 2007-09-05 | 05:30:00 |
| 3 | 1940-04-15 06:45:10 | 1940-04-15 | 21:45:10 |
| 4 | 2016-02-10 12:26:40 | 2016-02-11 | 04:26:40 |

In more detail:

```
dd %>%  
  mutate(oz = with_tz(dt, "Australia/Sydney")) %>%  
  pull(oz)
```

```
[1] "1970-01-01 22:50:01 AEST" "2007-09-05 05:30:00 AEST"  
[3] "1940-04-15 21:45:10 AEST" "2016-02-11 04:26:40 AEDT"
```

## How long between date-times?

- ▶ We may need to calculate the time between two events. For example, these are the dates and times that some patients were admitted to and discharged from a hospital:

```
admit,discharge
```

```
1981-12-10 22:00:00,1982-01-03 14:00:00
```

```
2014-03-07 14:00:00,2014-03-08 09:30:00
```

```
2016-08-31 21:00:00,2016-09-02 17:00:00
```

## Do they get read in as date-times?

- ▶ These ought to get read in and converted to date-times:

```
my_url <- "http://ritsokiguess.site/datafiles/hospital.csv"
stays <- read_csv(my_url)
stays
```

```
# A tibble: 3 x 2
```

|   | admit               |  | discharge           |  |
|---|---------------------|--|---------------------|--|
|   | <dtm>               |  | <dtm>               |  |
| 1 | 1981-12-10 22:00:00 |  | 1982-01-03 14:00:00 |  |
| 2 | 2014-03-07 14:00:00 |  | 2014-03-08 09:30:00 |  |
| 3 | 2016-08-31 21:00:00 |  | 2016-09-02 17:00:00 |  |

- ▶ and so it proves.

## Subtracting the date-times

- ▶ In the obvious way, this gets us an answer:

```
stays %>% mutate(stay = discharge - admit)
```

```
# A tibble: 3 x 3
```

|   | admit               |  | discharge           |  | stay        |
|---|---------------------|--|---------------------|--|-------------|
|   | <dtm>               |  | <dtm>               |  | <drtn>      |
| 1 | 1981-12-10 22:00:00 |  | 1982-01-03 14:00:00 |  | 568.0 hours |
| 2 | 2014-03-07 14:00:00 |  | 2014-03-08 09:30:00 |  | 19.5 hours  |
| 3 | 2016-08-31 21:00:00 |  | 2016-09-02 17:00:00 |  | 44.0 hours  |

- ▶ Number of hours; hard to interpret.

# Days

- ▶ Fractional number of days would be better:

```
stays %>%  
  mutate(  
    stay_days = as.period(admit %--% discharge) / days(1))
```

```
# A tibble: 3 x 3
```

|   | admit               |  | discharge           |  | stay_days |
|---|---------------------|--|---------------------|--|-----------|
|   | <dtm>               |  | <dtm>               |  | <dbl>     |
| 1 | 1981-12-10 22:00:00 |  | 1982-01-03 14:00:00 |  | 23.7      |
| 2 | 2014-03-07 14:00:00 |  | 2014-03-08 09:30:00 |  | 0.812     |
| 3 | 2016-08-31 21:00:00 |  | 2016-09-02 17:00:00 |  | 1.83      |



## Completed days

- ▶ Pull out with `day()` etc, as for a date-time:

```
stays %>%  
  mutate(  
    stay = as.period(admit %--% discharge),  
    stay_days = day(stay),  
    stay_hours = hour(stay)  
  ) %>%  
  select(starts_with("stay"))
```

```
# A tibble: 3 x 3
```

|   | stay          | stay_days | stay_hours |
|---|---------------|-----------|------------|
|   | <Period>      | <dbl>     | <dbl>      |
| 1 | 23d 16H 0M 0S | 23        | 16         |
| 2 | 19H 30M 0S    | 0         | 19         |
| 3 | 1d 20H 0M 0S  | 1         | 20         |

## Comments

- ▶ Date-times are stored internally as seconds-since-something, so that subtracting two of them will give, internally, a number of seconds.
- ▶ Just subtracting the date-times is displayed as a time (in units that R chooses for us).
- ▶ Convert to fractional times via a “period”, then divide by `days(1)`, `months(1)` etc.
- ▶ These ideas useful for calculating time from a start point until an event happens (in this case, a patient being discharged from hospital).