

Messy dates & times : : CHEAT SHEET

Retain, represent, and reason about date/time imprecision — an implementation of ISO 8601-2:2019 (EDTF) for R.



📅 Constructing messy dates

Anatomy of a messy date

One `mdate` value holds a date (and potentially time) in a standard layout:

```
2012-09-12 14:30:00
yyyy-mm-dd hh:mm:ss
```

Each component (y, m, d, h, m, s) or the whole date-time can be annotated:

Unspecified component (X)	2012-XX-12
Uncertain date-time (? suffixed)	2012-09-12 14:30?
Approximate time only (~ prefixed)	2012-09-12 ~14:30
Set of dates ({ } all, [] one-of)	{2012-09, 2012-11}
Range of dates (joined by ..)	2012-09..2012-10
Open ended: on or before/after	..2012-09-12

1. Annotate precise formats

```
      annotate
      as_messydate()
Date/POSIXct  --make_messydate()--> mdate
```

Tidy sources are precise; annotate information about messiness:

```
on_or_before("2010") -> "..2010"    on_or_after("2010") -> "2010.."
approximate("2010")   -> "2010~"    uncertain("2010") -> "2010?"
```

`component=` specifies part e.g.:

```
uncertain(x, component = "month") -> 2026-?07-07
```

2. Parse from (messy) text

```
"free text" --as_messydate()--> mdate
```

Messydates parser converts prose into annotated `mdate`:

```
"it happened about 44 BCE" -> "~0044"
"before 4th January 2024" -> "..2024-01-04"
"sometime in the 1980s" -> "198X"
"last day of Feb 2012" -> "2012-02-29"
"it was around 2:30pm" -> "14:30~"
```

See also `is_bce()`. `contract(md)` finds most concise messy form:

```
"2021-04-01..2021-04-30" -> "2021-04"
```

⚙ Working with messy dates

Adding, subtracting

Shift by days (numeric) or period strings, down to seconds:

```
"2012-01-31" + 1 -> "2012-02-01"
"2012-01-31 09:00" + "1 month" -> 2012-02-29 09:00
```

Calendar arithmetic retains day-of-month & time.

Combining two mdates

Finding multisets/ranges (with `=`, `-`, `%intersect%`, `%union%`) is easy:

```
"2012-01-31" + "2012-02-01" -> "2012-01-31..2012-02-01"
"2012-01" - "2012-01-31" -> "2012-01-01..2012-01-30"
"2012-01" %intersect% "2012-01-31" -> "2012-01-31"
```

See also `is_intersecting(x,y)`, `is_subset(x,y)`, `is_similar(x,y)`.

Durations: the `mduration` class

`make_messyduration(x, approx_range)` adds messy endpoints:

```
"2012-01", approx_range = 0 -> "2012-01-01..2012-01-31"
"2012-01", approx_range = 3 -> "2011-12-29..2012-02-03"
"2012-01", approx_range = -3 -> "2012-01-04..2012-01-28"
```

See also `seq(a, b, by = "hour")` to step through instants

Comparisons: exact or proportional

Exact comparisons (`<` `>` `<=` `>=`) answer only when certain:

```
"2012-06" < "2012-07-02" = TRUE, but
"2012-06" < "2012-06-02" = NA — not a false FALSE.
```

Proportional operators (`%l%`, `%le%`, `%g%`, `%ge%`, `%<%` within, `%>=%` within incl.) return *proportion* of x's dates passing test against y:

```
"2012-06" %l% "2012-06-02" ≈ 0.033
"2012-06" %le% "2012-06-02" ≈ 0.067
"2012-05..2012-06" %<% "2012-06-02" ≈ 0.016
```

Quantify precision

Precision of date-times in parts per day:

```
precision("2012") ≈ 0.003
precision("2012-01-01") = 1
precision("2012-01-01 14:30") = 1440
```

See also `is_precise` `is_uncertain` `is_approximate`

🔍 Analysing with messy dates

Coercion: expand, then resolve

`as.Date(mdate, FUN)`, `as.POSIXct(mdate, FUN)`, `as.POSIXlt(mdate, FUN)` coerce `mdate` to classes other packages expect for analysis.

Each coercion takes two steps:

- *expand* to all compatible dates, also available directly using `expand(mdate)`, then
- *resolve* them to a single date according to an explicit `FUN` rule

```
"2021-04" --expand()--> "2024-04-01"
              "2024-04-02" --FUN--> "2021-04-01"
              ...
```

`FUN` = makes the resolution rule an explicit, visible choice, and can be:

- *vectorised/element-wise* (`vmin` `vmax` `vmean` `vmedian` `vmodal`), or
- *summarise* the whole vector (`min` `max` `mean` `median` `modal` `random`)

```
vmin(c("2012", "2013")) -> "2012-01-01", "2013-01-01"
min(c("2012", "2013")) -> "2012-01-01"
vmax(c("2012", "2013")) -> "2012-12-31", "2013-12-31"
max(c("2012", "2013")) -> "2013-12-31"
vmean(c("2012", "2013")) -> "2012-07-01", "2013-07-02"
mean(c("2012", "2013")) -> "2012-12-31"
```

Precise values pass through untouched whatever the choice, and POSIX methods keep time-of-day.

Extracting components

{lubridate}-style accessors extract components of messy dates and return them as numerical values:

```
year() month() day() hour() minute() second() tz()
```

Describing & inferring across possibilities

Since resolution is an explicit step, varying it opens up opportunities to:

- describe according to `vmean/vmodal` most likely dates
- drop uncertain dates using `!is_uncertain`
- treat `vrandom` draws as multiple imputation for dates
- weight observations by precision
- bound result under `vmin` and `vmax`

See `vignette("messydates")` for a worked example.