

# Package ‘messydates’

July 21, 2022

**Title** A Flexible Class for Messy Dates

**Description** Contains a set of tools for constructing and coercing into and from the ```mdate` class.

This date class implements ISO 8601-2:2019(E) and allows regular dates to be annotated to express unspecified date components, approximate or uncertain date components, date ranges, and sets of dates.

This is useful for describing and analysing temporal information, whether historical or recent, where date precision may vary.

**Version** 0.3.1

**Date** 2022-07-21

**License** MIT + file LICENSE

**Encoding** UTF-8

**RoxygenNote** 7.2.0

**LazyData** true

**VignetteBuilder** knitr

**Imports** stringr, purrr, lubridate, tibble, dplyr

**Suggests** testthat (>= 3.0.0), covr, rmarkdown, knitr

**Config/testthat/edition** 3

**Depends** R (>= 3.4)

**NeedsCompilation** no

**Author** James Hollway [cre, aut, ctb] (IHEID, <<https://orcid.org/0000-0002-8361-9647>>),  
Henrique Sposito [ctb] (IHEID, <<https://orcid.org/0000-0003-3420-6085>>),  
Jael Tan [ctb] (IHEID, <<https://orcid.org/0000-0002-6234-9764>>)

**Maintainer** James Hollway <james.hollway@graduateinstitute.ch>

**Repository** CRAN

**Date/Publication** 2022-07-21 14:30:02 UTC

## R topics documented:

annotate . . . . .	2
battles . . . . .	3
class . . . . .	4
contract . . . . .	5
expand . . . . .	6
extract . . . . .	7
from_messydate . . . . .	8
is_messydate . . . . .	9
logical . . . . .	10
messydate . . . . .	10
mreport . . . . .	12
operate . . . . .	13
resolve . . . . .	13
set . . . . .	15
<b>Index</b>	<b>16</b>

---

annotate	<i>Annotates dates as censored, uncertain, or approximate</i>
----------	---

---

### Description

Some datasets have for example an arbitrary cut off point for start and end points, but these are often coded as precise dates when they are not necessarily the real start or end dates. This collection of functions helps annotate uncertainty and approximation to dates according to ISO2019E standards. Inaccurate start or end dates can be represented by an affix indicating "on or before", if used as a prefix (e.g. . . 1816-01-01), or indicating "on or after", if used as a suffix (e.g. 2016-12-31 . .). Approximate dates are indicated by adding a tilde to year, month, or day components, as well as groups of components or whole dates to estimate values that are possibly correct (e.g. 2003-03-03~). Day, month, or year, uncertainty can be indicated by adding a question mark to a possibly dubious date (e.g. 1916-10-10?) or date component (e.g. 1916-?10-10).

### Usage

```
on_or_before(x)
on_or_after(x)
as_approximate(x, component = NULL)
as_uncertain(x, component = NULL)
```

**Arguments**

x	A date vector
component	Annotation can be added on specific date components ("year", "month" or "day"), or to groups of date components (month and day ("md"), or year and month ("ym")). This must be specified. If unspecified, annotation will be added after the date (e.g. 1916-10-10?), indicating the whole date is uncertain or approximate. For specific date components, uncertainty or approximation is annotated to the left of the date component. E.g. for "day": 1916-10-?10 or 1916-10-~10. For groups of date components, uncertainty or approximation is annotated to the right of the group ("ym") or to both components ("md"). E.g. for "ym": 1916-10~-10; for "md": 1916-?10-?10.

**Value**

A mdate object with annotated date(s)

**Functions**

- on\_or\_before: prefixes dates with ".." where start date is uncertain
- on\_or\_after: suffixes dates with ".." where end date is uncertain
- as\_approximate: adds tildes to indicate approximate dates/date components
- as\_uncertain: adds question marks to indicate dubious dates/date components.

**Examples**

```
data <- data.frame(Beg = c("1816-01-01", "1916-01-01", "2016-01-01"),
  End = c("1816-12-31", "1916-12-31", "2016-12-31"))
dplyr::mutate(data, Beg = ifelse(Beg <= "1816-01-01",
  on_or_before(Beg), Beg))
dplyr::mutate(data, End = ifelse(End >= "2016-01-01",
  on_or_after(End), End))
dplyr::mutate(data, Beg = ifelse(Beg == "1916-01-01",
  as_approximate(Beg), Beg))
dplyr::mutate(data, End = ifelse(End == "1916-12-31",
  as_uncertain(End), End))
```

---

battles

*Dates of battles in 2001*


---

**Description**

A dataset containing the names and dates of battles in 2001, according to Wikipedia ([https://en.wikipedia.org/wiki/List\\_of\\_battles\\_in\\_2001](https://en.wikipedia.org/wiki/List_of_battles_in_2001))

**Usage**

```
battles
```

**Format**

A data frame with 20 rows and 2 variables:

**Battle** name of the battle, character

**Date** date or date range, a mdate class vector

**Parties** parties to the conflict, character

**Details**

```
## -----
## | Column Name | Data Type | Observations | Missing | Missing (%) |
## -----
## | Battle      | character | 20           | 0       | 0           |
## | Date        | mdate    | 20           | 0       | 0           |
## | Parties     | character | 20           | 0       | 0           |
## -----
```

---

class

*A flexible date class for messy dates*


---

**Description**

Recent extensions to standardised date notation in [ISO 8601-2\\_2019\(E\)](#) create space for unspecified, uncertain, and approximate dates, as well as succinct representation of ranges of dates. These functions create and validate a new date class for R that can contain and parse these annotations, and are not typically user-facing. Please see `as_messydate()` for the user-facing coercion function.

**Usage**

```
new_messydate(x = character())
```

```
validate_messydate(x)
```

```
NA_mdate_
```

**Arguments**

**x** A character scalar or vector in the expected "yyyy-mm-dd" format annotated, as necessary, according to ISO 8601-2\_2019(E).

**Format**

An object of class `mdate` of length 1.

## Details

### Date annotations:

*Unspecified date components*, such as when the day is unknown, can be represented by one or more Xs in place of the digits. The modifier \* is recommended to indicate that the entire time scale component value is unspecified, e.g. X\*-03-03, however this is not implemented here. Please be explicit about the digits that are unspecified, e.g. XXXX-03-03 expresses 3rd March in some unspecified year, whereas 2003-XX-03 expresses the 3rd of some month in 2003. If time components are not given, they are expanded to this.

*Approximate date components*, modified by ~, represent an estimate whose value is asserted to be possibly correct. For example, 2003~-03-03 The degree of confidence in approximation depends on the application.

*Uncertain date components*, modified by ?, represent a date component whose source is considered to be dubious and therefore not to be relied upon. An additional modifier, %, is used to indicate a value that is both uncertain and approximate.

### Date sets:

These functions also introduce standard notation for ranges of dates. Rather than the typical R notation for ranges, :, ISO 8601-2\_2019(E) recommends ... This then can be applied between two time scale components to create a standard range between these dates (inclusive), e.g. 2009-01-01..2019-01-01. But it can also be used as an affix, indicating "on or before" if used as a prefix, e.g. ..2019-01-01, or indicating "on or after" if used as a suffix, e.g. 2009-01-01..

And lastly, notation for sets of dates is also included. Here braces, {}, are used to mean "all members of the set", while brackets, [], are used to mean "one member of the set".

## Value

Object of class mdate

## See Also

messydate

---

contract

*Contract lists of dates into messy dates*

---

## Description

This function operates as the opposite of expand(). It contracts a list of dates into the abbreviated annotation of messy dates.

## Usage

```
contract(x, collapse = TRUE)
```

**Arguments**

x	A list of dates
collapse	Do you want ranges to be collapsed? TRUE by default. If FALSE ranges are returned in compact format.

**Value**

A mdate vector

**Examples**

```
d <- as_messydate(c("2001-01-01", "2001-01", "2001",
"2001-01-01..2001-02-02", "{2001-10-01,2001-10-04}",
"{2001-01,2001-02-02}", "28 BC", "-2000-01-01"))
e <- expand(d)
tibble::tibble(d, contract(e))
```

---

expand	<i>Expand messy dates to lists of dates</i>
--------	---

---

**Description**

These functions expand on date ranges, sets of dates, and unspecified or approximate dates (annotated with '.', ' ', 'XX' or '~'). As these messydates may refer to several possible dates, the function "opens" these values to reveal a vector of all the possible dates implied. Imprecise dates (dates only containing information on year and/or month) are also expanded to include possible dates within that year and/or month. The function removes the annotation from dates with unreliable sources ('?'), before being expanded normally as though they were incomplete.

**Usage**

```
expand(x, approx_range)

## S3 method for class 'mdate'
expand(x, approx_range = 0)
```

**Arguments**

x	A mdate object.
approx_range	Range to expand approximate dates, or date components, annotated with '~', by default 0. That is, removes signs for approximate dates and treats these dates as precise dates. If 3, for example, adds 3 days for day approximation, 3 months for month approximation, 3 years for year/whole date approximation, 3 years and 3 months for year-month approximation, and 3 months and 3 days for month-day approximation.

**Value**

A list of dates, including all dates in each range or set.

**Methods (by class)**

- mdate: Expanding messydates

**Examples**

```
d <- as_messydate(c("2008-03-25", "-2012-02-27", "2001-01?", "~2001",
"2001-01-01..2001-02-02", "{2001-01-01,2001-02-02}", "{2001-01,2001-02-02}",
"2008-XX-31", "..2002-02-03", "2001-01-03..", "28 BC"))
expand(d)
```

---

extract

*Extracting components from messy dates*

---

**Description**

These functions allow the extraction of particular date components from messy dates, such as the `year()`, `month()`, and `day()`. `precision()` allows for the identification of the greatest level of precision in (currently) the first element of each date.

**Usage**

```
year(x)
```

```
month(x)
```

```
day(x)
```

```
precision(x)
```

**Arguments**

x                    A mdate object

**Value**

`year()`, `month()`, and `day()` extraction return the integer for the requested date component. `precision()` returns the level of greatest precision for each date.

**Examples**

```
year(as_messydate(c("2012-02-03", "2012", "2012-02")))
month(as_messydate(c("2012-02-03", "2012", "2012-02")))
day(as_messydate(c("2012-02-03", "2012", "2012-02")))
precision(as_messydate(c("2012-02-03", "2012", "2012-02")))
```

---

from_messydate	<i>Coercion from messy dates</i>
----------------	----------------------------------

---

### Description

These functions coerce objects of `mdate` class to common date classes such as `Date`, `POSIXct`, and `POSIXlt`. Since `mdate` objects can hold multiple individual dates, however, an additional function must be passed as an argument so that these functions know how to coerce resolve multiple dates into a single date.

For example, one might wish to use the earliest possible date in any ranges of dates (`min`), the latest possible date (`max`), some notion of a central tendency (`mean`, `median`, or `modal`), or even a random selection from among the candidate dates.

These functions then, building on `expand()` and the resolve functions, are particularly useful in converting back out of the `mdate` class for use with existing methods and models, especially for checking the robustness of results.

### Usage

```
## S3 method for class 'mdate'
as.Date(x, ..., FUN)
```

```
## S3 method for class 'mdate'
as.POSIXct(x, ..., FUN)
```

```
## S3 method for class 'mdate'
as.POSIXlt(x, ..., FUN)
```

### Arguments

<code>x</code>	A <code>mdate</code> object
<code>...</code>	Arguments passed on to the S3 generics.
<code>FUN</code>	A function that can be used to resolve expanded messy dates into a single date. For example, <code>min()</code> , <code>max()</code> , <code>mean()</code> , <code>median()</code> , <code>modal()</code> , and <code>random()</code> .

### Value

A date object of `Date`, `POSIXct`, or `POSIXlt` class

### Examples

```
as.Date(as_messydate("2012-01"), min)
as.Date(as_messydate("2012-01-01"), mean)
as.Date(as_messydate("2012-01"), max)
as.Date(as_messydate("2012-01"), median)
as.Date(as_messydate("2012-01"), modal)
as.Date(as_messydate("2012-01"), random)
as.Date(as_messydate("1000 BC"), max)
```



```
as.Date(as_messydate("1000 BC"), mean)
as.Date(as_messydate("1000 BC"), median)
as.Date(as_messydate(c("-1000", "2020")), min)
```

---

is_messydate	<i>Logical tests on messy dates</i>
--------------	-------------------------------------

---

## Description

These functions provide various logical tests for messy date objects.

## Usage

```
is_messydate(x)
is_intersecting(x, y)
is_element(x, y)
is_similar(x, y)
is_precise(x)
is_uncertain(x)
is_approximate(x)
```

## Arguments

x, y                    mdate or other class objects

## Value

A logical vector the same length as the mdate passed.

## Functions

- `is_messydate`: tests whether the object inherits the mdate class. If more rigorous validation is required, see `validate_messydate()`.
- `is_intersecting`: tests whether there is any intersection between two messy dates, leveraging `intersect()`.
- `is_element`: tests whether a messy date can be found within a messy date range or set.
- `is_similar`: tests whether two dates contain similar components. This can be useful for identifying dates that may be typos of one another.
- `is_precise`: tests whether a date is precise (i.e. an 8 digit date). Non-precise dates contain markers that they are approximate (i.e. ~), unreliable (i.e. ?), are incomplete dates (i.e. year only), or date ranges and sets.

- `is_uncertain`: tests whether a date is uncertain (i.e. contains ?).
- `is_approximate`: tests whether a date is approximate (i.e. contains ~).

### Examples

```
is_messydate(as_messydate("2012-01-01"))
is_messydate(as.Date("2012-01-01"))
is_intersecting(as_messydate("2012-01"),
as_messydate("2012-01-01..2012-02-22"))
is_intersecting(as_messydate("2012-01"),
as_messydate("2012-02-01..2012-02-22"))
is_element(as_messydate("2012-01-01"), as_messydate("2012-01"))
is_element(as_messydate("2012-01-01"), as_messydate("2012-02"))
is_similar(as_messydate("2012-06-02"), as_messydate("2012-02-06"))
is_similar(as_messydate("2012-06-22"), as_messydate("2012-02-06"))
is_precise(as_messydate(c("2012-06-02", "2012-06")))
is_uncertain(as_messydate(c("2012-06-02", "2012-06-02?")))
is_approximate(as_messydate(c("2012-06-02~", "2012-06-02")))
```

---

logical

*Logical tests on messy dates*

---

### Description

These functions provide various logical tests for messy date objects.

### Arguments

`x, y`                    `mdate` or other class objects

### Value

A logical vector the same length as the `mdate` passed.

---

messydate

*Coercion from regular date classes to mdate*

---

### Description

These methods coerce various date classes into the `mdate` class. They represent the main user-facing class-creating functions in the package. In addition to the typical date classes in R (`Date`, `POSIXct`, and `POSIXlt`), there is also a direct method for converting text or character strings to `mdate`. The function can also extract dates from text, though this is a work-in-progress and currently only works in English.

**Usage**

```

as_messydate(x, resequence = FALSE)

## S3 method for class 'Date'
as_messydate(x, resequence = FALSE)

## S3 method for class 'POSIXct'
as_messydate(x, resequence = FALSE)

## S3 method for class 'POSIXlt'
as_messydate(x, resequence = FALSE)

## S3 method for class 'character'
as_messydate(x, resequence = FALSE)

make_messydate(..., resequence = FALSE)

```

**Arguments**

x	A scalar or vector of a class that can be coerced into mdate, such as Date, POSIXct, POSIXlt, or character.
resequence	Users have the option to choose the order for ambiguous 6 digit dates (e.g. "11-01-12"), and to expand these dates into precise dates (i.e. YYYY-MM-DD format). FALSE by default. If TRUE, it prompts users to select the existing component order of ambiguous 6 digit dates, based on which the date is reordered into YY-MM-DD format and further completed to YYYY-MM-DD format if they choose to do so.
...	One (yyyy-mm-dd) or three (yyyy, mm, dd) variables

**Value**

A mdate class object

**Functions**

- `as_messydate`: Core mdate class coercion function
- `as_messydate.Date`: Coerce from Date to mdate class
- `as_messydate.POSIXct`: Coerce from POSIXct to mdate class
- `as_messydate.POSIXlt`: Coerce from POSIXlt to mdate class
- `as_messydate.character`: Coerce character date objects to mdate class
- `make_messydate`: Composes mdate from multiple variables

**Examples**

```

as_messydate("2021")
as_messydate("2021-02")
as_messydate("2021-02-01")

```

```
as_messydate("01-02-2021")
as_messydate("1 February 2021")
as_messydate("First of February, two thousand and twenty-one")
as_messydate("2021-02-01?")
as_messydate("2021-02-01~")
as_messydate("2021-02-01%")
as_messydate("2021-02-01..2021-02-28")
as_messydate("{2021-02-01,2021-02-28}")
as_messydate(c("-2021", "2021 BC", "-2021-02-01"))
# as_messydate(c("01-02-21", "01-02-2021", "01-02-91", "01-02-1991"),
# resequence = TRUE)
make_messydate("2010", "10", "10")
```

---

mreport

*Data report for datasets with 'mdate' variables*

---

## Description

Create a properly formatted data report for datasets which contain 'mdate' class objects, alongside other object classes.

## Usage

```
mreport(data)
```

## Arguments

data            A {tibble} or a {data.frame}.

## Details

'mreport' displays the variable's name, the variable type, the number of observations per variable, the number of missing observations for variable, and the percentage of missing observations in variable.

## Value

A data report of class 'mreport'.

## Examples

```
mreport(battles)
```

---

operate	<i>Arithmetic operations for messydates</i>
---------	---

---

### Description

These operations allow users to add or subtract dates messydate objects. Messydate objects include incomplete or uncertain dates, ranges of dates, negative dates, and date sets.

### Usage

```
## S3 method for class 'mdate'
e1 + e2

## S3 method for class 'mdate'
e1 - e2
```

### Arguments

e1	A messydate object
e2	A numerical object.

### Value

A messydates vector

### Examples

```
d <- as_messydate(c("2008-03-25", "-2012-02-27", "2001-01?", "~2001",
"2001-01-01..2001-02-02", "{2001-01-01,2001-02-02}",
"2008-XX-31", "..2002-02-03", "2001-01-03..", "28 BC"))
tibble::tibble(date = d, add = d + 1, subtract = d - 1)
tibble::tibble(date = d, add = d + "1 year", subtract = d - "1 year")
```

---

resolve	<i>Resolves messy dates into a single value</i>
---------	---

---

### Description

This collection of S3 methods 'resolve' messy dates into a single date according to some explicit bias, such as returning the minimum or maximum date, the mean, median, or modal date, or a random date from among the possible resolutions for each messy date. If the date is not 'messy' (i.e. has no annotations) then just that precise date is returned. This can be useful for various descriptive or inferential projects.

**Usage**

```
## S3 method for class 'mdate'
min(..., na.rm = TRUE)

## S3 method for class 'mdate'
max(..., na.rm = TRUE)

## S3 method for class 'mdate'
median(..., na.rm = TRUE)

## S3 method for class 'mdate'
mean(..., trim = 0, na.rm = TRUE)

modal(..., na.rm = FALSE)

## S3 method for class 'mdate'
modal(..., na.rm = TRUE)

random(..., size, replace = FALSE, prob = NULL)

## S3 method for class 'mdate'
random(..., size, replace = FALSE, prob = NULL)
```

**Arguments**

...	a mdate object
na.rm	Should NAs be removed? True by default.
trim	the fraction (0 to 0.5) of observations to be trimmed from each end of x before the mean is computed. Values of trim outside that range are taken as the nearest endpoint.
size	a non-negative integer giving the number of items to choose.
replace	should sampling be with replacement?
prob	a vector of probability weights for obtaining the elements of the vector being sampled.

**Value**

A single scalar or vector of dates

**Examples**

```
d <- as_messydate(c("2008-03-25", "?2012-02-27", "2001-01?", "2001~",
"2001-01-01..2001-02-02", "{2001-01-01,2001-02-02}",
"{2001-01,2001-02-02}", "2008-XX-31"))
d
min(d)
max(d)
mean(d)
```

```
median(d)
modal(d)
random(d)
```

---

set

*Set operations for messy dates*

---

### Description

Performs intersection (`md_intersect()`) and union (`md_union()`) on, inter alia, messy date class objects. For a more typical 'join' that retains all elements, even if duplicated, please use `md_multiset`.

### Usage

```
md_intersect(...)

md_union(x, y)

md_multiset(x, y)
```

### Arguments

`x, y, ...` Messy date or other class objects

### Value

A vector of the same mode for intersect, or a common mode for union.

### Functions

- `md_intersect`: Find intersection of sets of messy dates
- `md_union`: Find union of sets of messy dates
- `md_multiset`: Join two sets of messy dates

### Examples

```
md_intersect(as_messydate("2012-01-01..2012-01-20"), as_messydate("2012-01"))
md_union(as_messydate("2012-01-01..2012-01-20"), as_messydate("2012-01"))
md_multiset(as_messydate("2012-01-01..2012-01-20"), as_messydate("2012-01"))
```

# Index

## \* datasets

- battles, 3
- class, 4
- + .mdate (operate), 13
- .mdate (operate), 13
- annotate, 2
- as.Date.mdate (from\_messydate), 8
- as.POSIXct.mdate (from\_messydate), 8
- as.POSIXlt.mdate (from\_messydate), 8
- as\_approximate (annotate), 2
- as\_messydate (messydate), 10
- as\_uncertain (annotate), 2
- battles, 3
- class, 4
- contract, 5
- day (extract), 7
- expand, 6
- extract, 7
- from\_messydate, 8
- is\_approximate (is\_messydate), 9
- is\_element (is\_messydate), 9
- is\_intersecting (is\_messydate), 9
- is\_messydate, 9
- is\_precise (is\_messydate), 9
- is\_similar (is\_messydate), 9
- is\_uncertain (is\_messydate), 9
- logical, 10
- make\_messydate (messydate), 10
- max.mdate (resolve), 13
- md\_intersect (set), 15
- md\_multiset (set), 15
- md\_union (set), 15
- mean.mdate (resolve), 13
- median.mdate (resolve), 13
- messydate, 10
- min.mdate (resolve), 13
- modal (resolve), 13
- month (extract), 7
- mreport, 12
- NA\_mdate\_ (class), 4
- new\_messydate (class), 4
- on\_or\_after (annotate), 2
- on\_or\_before (annotate), 2
- operate, 13
- precision (extract), 7
- random (resolve), 13
- resolve, 13
- set, 15
- validate\_messydate (class), 4
- year (extract), 7