Package: messydates (via r-universe)

March 7, 2025

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 0.5.2

Date 2025-03-07

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

LazyData true

Depends R (>= 4.0)

Imports stringi, purrr, lubridate, dplyr

Suggests testthat (>= 3.0.0), rmarkdown

URL https://globalgov.github.io/messydates/

BugReports https://github.com/globalgov/messydates/issues

Config/Needs/build roxygen2, devtools

Config/Needs/check covr, lintr, spelling

Config/Needs/website pkgdown

Config/testthat/edition 3

Config/testthat/parallel true

Config/testthat/start-first logical-operators

Config/pak/sysreqs libicu-dev

Repository https://globalgov.r-universe.dev

RemoteUrl https://github.com/globalgov/messydates

RemoteRef HEAD

RemoteSha 0acaaa02a0bc0e3c68f7ec09315b3d63a545e0dc

Contents

`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_ba

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

US_party is the US a party to the battle, numeric

N_actors number of actors to conflict, numeric

`class_create` *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)
```

Arguments

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

Value

Object of class `mdate`

Date annotations

Unspecified date components, such as when the day is unknown, can be represented by one or more `X`s 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".

See Also

messydate

`class_duration` *A duration class for mdates*

Description

The `mdates_duration` class introduces methods that annotate a duration or period with representations of its uncertainty.

Usage

```
new_messyduration(x = character())

messyduration(x, approx_range = 0)

validate_messyduration(x, approx_range = 0)

## S3 method for class 'character'
messyduration(x, approx_range = 0)

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

Arguments

`x`	An `mdate` variable with ranges.
`approx_range`	Range to expand approximate dates, in days. If 3, for example, adds 3 days; if -3, removes 3 days from both sides.

Details

Most R packages handle duration and periods as exact time or date intervals. However, this is not possible for 'messy' dates where uncertainty or approximation might be present. The `mdates_duration` class accounts for uncertainty and approximation in `mdate` objects to return their duration as a range of possible dates.

Value

Object of class `description`

Examples

```
messyduration(as_messydate(c("2010-01-01..2010-12-31", "2010-01..2010-12")))
```

class_make *Composes* mdate *from multiple variables*

Description

Composes mdate from multiple variables

Usage

```
make_messydate(..., resequence = FALSE)
```

Arguments

...	One (yyyy-mm-dd), two (yyyy-mm-dd, yyyy-mm-dd), or three (yyyy, mm, dd) variables.
resequence	Users have the option to choose the order for ambiguous dates with or without separators (e.g. "11-01-12" or "20112112"). NULL by default. Other options include: 'dmy', 'ymd', 'mdy', 'ym', 'my' and 'interactive' If 'dmy', dates are converted from DDMMYY format for 6 digit dates, or DDMMYYYY format for 8 digit dates. If 'ymd', dates are converted from YYMMDD format for 6 digit dates, or YYYYMMDD format for 8 digit dates. If 'mdy', dates are converted from MMDDYY format for 6 digit dates or MMDDYYYY format for 8 digit dates. For these three options, ambiguous dates are converted to YY-MM-DD format for 6 digit dates, or YYYY-MM-DD format for 8 digit dates. If 'my', ambiguous 6 digit dates are converted from MM-YYYY format to YYYY-MM. If 'ym', ambiguous 6 digit dates are converted to YYYY-MM format. If 'interactive', it prompts users to select the existing component order of ambiguous dates, based on which the date is reordered into YYYY-MM-DD format and further completed to YYYY-MM-DD format if they choose to do so.

Details

If three date variables are passed to make_messydate(), function will create a single date (yyyy-mm-dd) from it. If two date variables are passed to make_messydate(), function will create a range of dates from it (yyyy-mm-dd..yyyy-mm-dd). If one date variable is passed to make_messydate(), function defaults to as_messydate().

Examples

```
make_messydate("2010", "10", "10")
```

coerce_extrema *Resolves messy dates into an extrema*

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

```
vmin(..., na.rm = FALSE)

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

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

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

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

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

Arguments

...	a mdate object
na.rm	Should NAs be removed? True by default.

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", "-0050-01-01"))
d
vmin(d)
min(d)
vmax(d)
max(d)
```

coerce_from *Coercion from messy dates*

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 = vmin, ...)

## S3 method for class 'mdate'
as.POSIXct(x, tz = "UTC", FUN = vmin, ...)

## S3 method for class 'mdate'
as.POSIXlt(x, tz = "UTC", FUN = vmin, ...)
```

Arguments

x	A mdate object
FUN	A function that can be used to resolve expanded messy dates into a single date. For example, min(), max(), mean(), median(), modal(), and random().
...	Arguments passed on to the S3 generics.
tz	Character string specifying the time zone for the conversion, if required. By default "UTC" (Universal Time Coordinated), equivalent to GMT. If "" then the current time zone is used.

Value

A date object of Date, POSIXct, or POSIXlt class

Examples

```
as.Date(as_messydate("2012-01"), FUN = vmin)
as.Date(as_messydate("2012-01-01"), FUN = vmean)
as.Date(as_messydate("2012-01"), FUN = vmax)
```

```
as.Date(as_messydate("2012-01"), FUN = vmedian)
as.Date(as_messydate("2012-01"), FUN = vmodal)
as.Date(as_messydate("2012-01"), FUN = vrandom)
as.Date(as_messydate("1000 BC"), FUN = vmax)
as.Date(as_messydate("1000 BC"), FUN = vmedian)
as.Date(as_messydate(c("-1000", "2020")), FUN = vmin)
```

`coerce_tendency` *Resolves messy dates into a central tendency*

Description

These functions resolve messydates by their central tendency. While the functions `mean()`, `median()`, and `modal()` summarise the vector to a single value, `v*()` versions return a vector of the same length.

Usage

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

vmedian(..., na.rm = TRUE)

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

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

vmean(..., na.rm = TRUE)

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

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

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

vmodal(..., na.rm = TRUE)

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

random(..., na.rm = TRUE)

## S3 method for class 'mdate'
```

```
random(..., na.rm = TRUE)

vrandom(..., na.rm = TRUE)

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

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.

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", "-0050-01-01"))
d
median(d)
vmedian(d)
mean(d)
vmean(d)
modal(d)
vmodal(d)
random(d)
vrandom(d)
```

---

coerce_to *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 = NULL)

## S3 method for class 'numeric'
as_messydate(x, resequence = NULL)

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

mdate(x, 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 dates with or without separators (e.g. "11-01-12" or "20112112"). NULL by default. Other options include: 'dmy', 'ymd', 'mdy', 'ym', 'my' and 'interactive' If 'dmy', dates are converted from DDMMYY format for 6 digit dates, or DDMMYYYY format for 8 digit dates. If 'ymd', dates are converted from YYMMDD format for 6 digit dates, or YYYYMMDD format for 8 digit dates. If 'mdy', dates are converted from MMDDYY format for 6 digit dates or MMDDYYYY format for 8 digit dates. For these three options, ambiguous dates are converted to YY-MM-DD format for 6 digit dates, or YYYY-MM-DD format for 8 digit dates. If 'my', ambiguous 6 digit dates are converted from MM-YYYY format to YYYY-MM. If 'ym', ambiguous 6 digit dates are converted to YYYY-MM format. If 'interactive', it prompts users to select the existing component order of ambiguous dates, based on which the date is reordered into YYYY-MM-DD format and further completed to YYYY-MM-DD format if they choose to do so.

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

- `as_messydate(numeric)`: Coerce numeric objects to `mdate` class
- `as_messydate(list)`: Coerce list date objects to the most concise representation of `mdate` class

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("210201", "20210201"), resequence = "ymd")
as_messydate(c("010221", "01022021"), resequence = "dmy")
# as_messydate(c("01-02-21", "01-02-2021", "01-02-91", "01-02-1991"),
# resequence = "interactive")
as_messydate(list(c("2012-06-01", "2012-06-02", "2012-06-03")))
as_messydate(list(c("2012-06-01", "2012-06-02", "2012-06-03",
"{2012-06-01, 2012-06-02, 2012-06-03}", "2012-06-01", "2012-06-03")))
```

---

component_annotate *Annotates dates as censored, uncertain, or approximate*

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))
```

---

component_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")))
```

---

convert_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.

Details

The ´contract()´ function first expand() 'mdate' objects to then display their most succinct representation.

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",
"{2001-01-01, 2001-01-02, 2001-01-03}"))
dplyr::tibble(d, contract(d))
```

---

convert_expand *Expand messy dates to lists of dates*

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 = 0)
```

Arguments

x	A mdate object. If not an 'mdate' object, conversion is handled first with ´as_messydate()´.
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.

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)
```

convert_sequence *Sequence method for messydates*

Description

This function provides a sequence (`seq()`) method for messydates. This can be used with ranges or unspecified dates, and is particularly useful for defining a sequence of dates before the common era or between eras.

Usage

```
## S3 method for class 'mdate'
seq(from, to, by = "days", ...)
```

Arguments

`from`	A messydate or range. If 'from' is a range and 'to' is not specified, 'from' will be the minimum of the range and 'to' will be maximum.
`to`	A messydate.
`by`	Increment of the sequence. By default "days".
`...`	Arguments passed to or from methods.

Examples

```
seq(mdate("-0001-12-20"), mdate("0001-01-10"))
```

operate_arithmetic *Arithmetic operations for messydates*

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`	An `mdate` or date object.
`e2`	An `mdate`, date, or numeric object. Must be a scalar.

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"))
dplyr::tibble(date = d, add = d + 1, subtract = d - 1)
dplyr::tibble(date = d, add = d + "1 year", subtract = d - "1 year")
as_messydate("2001-01-01") + as_messydate("2001-01-02..2001-01-04")
as_messydate("2001-01-01") + as_messydate("2001-01-03")
as_messydate("2001-01-01..2001-01-04") - as_messydate("2001-01-02")
#as_messydate("2001-01-01") - as_messydate("2001-01-03")
```

---

operate_inequalities *Logical operations on messy dates*

Description

Logical operations on messy dates

Usage

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

## S3 method for class 'mdate'
e1 > e2

## S3 method for class 'mdate'
e1 <= e2

## S3 method for class 'mdate'
e1 >= e2
```

Arguments

e1, e2	mdate or other class objects

Functions

- `<`: tests whether the dates in the first vector precede the dates in the second vector. Returns NA when the date order can't be determined.
- `>`: tests whether the dates in the first vector succeed the dates in the second vector. Returns NA when the date order can't be determined.

- <= : tests whether the dates in the first vector are equal to or precede the dates in the second vector. Returns NA when the date order can't be determined.
- >= : tests whether the dates in the first vector are equal to or succeed the dates in the second vector. Returns NA when the date order can't be determined.

Examples

```
as_messydate("2012-06-02") > as.Date("2012-06-01") # TRUE
# 2012-06-XX could mean 2012-06-03, so unknown if it comes before 2012-06-02
as_messydate("2012-06-XX") < as.Date("2012-06-02") # NA
# But 2012-06-XX cannot be before 2012-06-01
as_messydate("2012-06-XX") >= as.Date("2012-06-01") # TRUE
```

---

operate_proportional *Proportion of messy dates meeting logical test*

Description

These functions provide various proportional tests for messy date objects.

Usage

```
e1 %l% e2

## S3 method for class 'mdate'
e1 %l% e2

e1 %g% e2

## S3 method for class 'mdate'
e1 %g% e2

e1 %ge% e2

## S3 method for class 'mdate'
e1 %ge% e2

e1 %le% e2

## S3 method for class 'mdate'
e1 %le% e2

e1 %><% e2

## S3 method for class 'mdate'
e1 %><% e2
```

```
e1 %>=<% e2

## S3 method for class 'mdate'
e1 %>=<% e2
```

Arguments

e1, e2	mdate or other class objects

Value

The proportion that the comparison is true.

A logical vector the same length as the mdate passed.

Functions

- %l% : Tests proportion of dates in the first vector that precede the minimum in the second vector.
- %g% : Tests proportion of dates in the first vector that follow the maximum in the second vector.
- %ge% : Tests proportion of dates in the first vector that follow or are equal to the maximum in the second vector.
- %le% : Tests proportion of dates in the first vector that precede or are equal to the minimum in the second vector.
- %><% : Tests proportion of dates in the first vector that are between the minimum and maximum dates in the second vector.
- %>=<% : Tests proportion of dates in the first vector that are between the minimum and maximum dates in the second vector, inclusive.

Examples

```
as_messydate("2012-06") < as.Date("2012-06-02")
as_messydate("2012-06") %l% as_messydate("2012-06-02")
as_messydate("2012-06") > as.Date("2012-06-02")
as_messydate("2012-06") %g% as_messydate("2012-06-02")
as_messydate("2012-06") >= as.Date("2012-06-02")
as_messydate("2012-06") %ge% as_messydate("2012-06-02")
as_messydate("2012-06") <= as.Date("2012-06-02")
as_messydate("2012-06") %le% "2012-06-02"
as_messydate("2012-06") %><% as_messydate("2012-06-15..2012-07-15")
as_messydate("2012-06") %>=<% as_messydate("2012-06-15..2012-07-15")
```

operate_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

```
e1 %intersect% e2

## S3 method for class 'mdate'
e1 %intersect% e2

e1 %union% e2

## S3 method for class 'mdate'
e1 %union% e2
```

Arguments

`e1`, `e2`	Messy date or other class objects

Value

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

Functions

- `%intersect%` : Find intersection of sets of messy dates
- `%union%` : Find intersection of sets of messy dates

Examples

```
as_messydate("2012-01-01..2012-01-20") %intersect% as_messydate("2012-01")
as_messydate("2012-01-01..2012-01-20") %union% as_messydate("2012-01")
```

operate_statements *Logical statements on messy dates*

Description

These functions provide various logical statements about messy date objects.

Usage

```
is_messydate(x)

is_intersecting(x, y)

is_subset(x, y)

is_similar(x, y)

is_precise(x)

is_uncertain(x)

is_approximate(x)

is_bce(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_subset(): tests whether one or more 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 ~).
- `is_bce()`: tests whether one or more messy dates are found before the common era.

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_subset(as_messydate("2012-01-01"), as_messydate("2012-01"))
is_subset(as_messydate("2012-01-01..2012-01-03"), as_messydate("2012-01"))
is_subset(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")))
is_bce(as_messydate(c("2012-06-02", "-2012-06-02")))
```

Index