Package 'messydates'

July 19, 2021

Title A Flexible Class for Messy Dates

Description Contains a set of tools for constructing and coercing
into and from the messydt 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 0.1.1

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.1.1

Imports covr, stringr, purrr, lubridate, tibble, dplyr

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

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>)

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

Repository CRAN

Date/Publication 2021-07-19 07:50:02 UTC

R topics documented:

`as_messydate` *Coercion to messy dates*

Description

These functions coerce different data classes into `messydt` class

Usage

```
as_messydate(x)

## S3 method for class 'Date'
as_messydate(x)

## S3 method for class 'POSIXct'
as_messydate(x)

## S3 method for class 'POSIXlt'
as_messydate(x)

## S3 method for class 'character'
as_messydate(x)
```

Arguments

`x`	A scalar or vector of a class that can be coerced into a Date, such as `Date`, `POSIXct`, `POSIXlt`, or character.

Value

A `messydt` class object

Methods (by class)

- `Date`: Coerce from `Date` to `messydt` class
- `POSIXct`: Coerce from `POSIXct` to `messydt` class
- `POSIXlt`: Coerce from `POSIXlt` to `messydt` class
- `character`: Coerce character date objects to `messydt` class

Examples

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

`class` *A broader date class for messy dates*

Description

These functions create and validate a new date class for R consistent with ISO 8601-2_2019(E). These recent extensions to standardised date notation create space for unspecified, uncertain, and approximate dates, as well as succinct representation of date ranges.

Usage

```
new_messydate(x = character())

validate_messydate(x)

NA_messydt_
```

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 `messydt` of length 1.

Value

Object of class `messydt`

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

as_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 = list())
```

Arguments

x	A list of dates

Value

A messydt vector

Examples

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

expand *Expand messy dates to lists of dates*

Description

These functions expand on unspecified dates, ranges and on sets of dates. Uncertain dates may include several possible dates. The function "opens" these values to include all the possible dates contained in uncertain dates.

Usage

```
expand(x)

## S3 method for class 'messydt'
expand(x)
```

Arguments

x	A messydt object.

Value

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

Methods (by class)

- messydt: Expanding messydates

Examples

```
d <- as_messydate(c("2001-01-01", "2001-01", "2001", "2001-01-01..2001-02-02",
        "{2001-01-01,2001-02-02}", "{2001-01,2001-02-02}"))
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(md)

month(md)

day(md)

precision(md)
```

Arguments

md	A messy date 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 *Coercion from messy dates*

Description

These functions coerce objects of `messydt` class to common date classes such as `Date`, `POSIXct`, and `POSIXlt`. Since `messydt` 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 amongst the candidate dates.

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

Usage

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

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

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

Arguments

x	A messydt object
...	Arguments passed on to the S3 generics.
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().

Value

A date object of Date, POSIXct, or POSIXlt class

Examples

```
as.Date(as_messydate("2012-01"), min)
as.Date(as_messydate("2012-01"), 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)
```

---

logical *Logical tests on messy dates*

Description

These functions provide various logical tests for messy date objects. is_messydate() tests whether the object inherits the messydt class. If a more rigorous validation is required, see validate_messydate(). is_intersecting() tests whether there is any intersection between two messy dates, leveraging intersect(). is_element() similarly 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.

Usage

```
is_messydate(x)

is_intersecting(x, y)

is_element(x, y)

is_similar(x, y)
```

Arguments

x, y	Messy date or other class objects

Value

A logical vector the length of the messy dates passed.

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"))
```

---

make_messydate *Make messy dates from multiple variables*

Description

Transforms multiple date inputs contained in different columns into one.

Usage

```
make_messydate(...)
```

Arguments

...	One (ymd) or three (yyyy, mm, dd) variables

Value

A character vector containing the

Examples

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

`resolve` *Resolves messy dates into a single value*

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 'messydt'
min(..., na.rm = TRUE)

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

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

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

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

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

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

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

Arguments

`...`	a messydt 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("2014-01-01..2014-01-31")
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