Package 'aweek'

March 9, 2019

Title Convert Dates to Arbitrary Week Definitions

Version 0.2.0

Description Which day a week starts depends heavily on the either the local or professional context. This package is designed to be a lightweight solution to easily switching between week-based date definitions.

Depends R (>= 3.0)

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Suggests testthat, roxygen2, knitr, covr

RoxygenNote 6.1.1

URL https://github.com/reconhub/aweek

BugReports https://github.com/reconhub/aweek/issues

VignetteBuilder knitr

NeedsCompilation no

Author Zhian N. Kamvar [aut, cre]

Maintainer Zhian N. Kamvar <zkamvar@gmail.com>

Repository CRAN

Date/Publication 2019-03-08 23:12:31 UTC

R topics documented:

as.POSIXlt.aweek 2
date2week 3
print.aweek 5

Index **7**

`as.POSIXlt.aweek` *Convert aweek objects to characters or dates*

Description

Convert aweek objects to characters or dates

Usage

```
## S3 method for class 'aweek'
as.POSIXlt(x, tz = "", floor_day = FALSE, ...)

## S3 method for class 'aweek'
as.character(x, ...)

## S3 method for class 'aweek'
as.Date(x, floor_day = FALSE, ...)
```

Arguments

`x`	an object of class aweek.
`tz`	passed on to `as.POSIXlt()`
`floor_day`	when `TRUE`, the days will be set to the start of the week.
`...`	parameters passed to `as.POSIXlt()`.

See Also

`date2week()` `print.aweek()`

Examples

```
w <- date2week(Sys.Date(), week_start = "Sunday")
w
# convert to POSIX
as.POSIXlt(w)
as.POSIXlt(w, floor_day = TRUE)
as.POSIXlt(w, floor_day = TRUE, tz = "KST")

# convert to date
as.Date(w)
as.Date(w, floor_day = TRUE)

# convert to character (strip attributes)
as.character(w)
```

`date2week` *Convert date to a an arbitrary week definition*

Description

Convert date to a an arbitrary week definition

Usage

```
date2week(x, week_start = 1, floor_day = FALSE, numeric = FALSE,
  factor = FALSE, ...)

week2date(x, week_start = 1, floor_day = FALSE)
```

Arguments

`x`	a Date, POSIXt, character, or any data that can be easily converted to a date with `as.POSIXlt()`.
`week_start`	a number indicating the start of the week based on the ISO 8601 standard from 1 to 7 where 1 = Monday OR an abbreviation of the weekdate in an English or current locale. *Note: using a non-English locale may render your code non-portable.*
`floor_day`	when `TRUE`, the days will be set to the start of the week.
`numeric`	if `TRUE`, only the numeric week be returned. If `FALSE` (default), the date in the format "YYYY-Www-d" will be returned.
`factor`	if `TRUE`, a factor will be returned with levels spanning the range of dates. If `floor_date = FALSE`, then this will use the sequence of days between the first and last date, but if `floor_date = TRUE`, then the sequence of weeks between the first and last date will be used. *Take caution when using this with a large date range as the resulting factor can contain all days between dates.*
`...`	arguments passed to `as.POSIXlt()`, unused in all other cases.

Details

Weeks differ in their start dates depending on context. The ISO 8601 standard specifies that Monday starts the week (https://en.wikipedia.org/wiki/ISO_week_date) while the US CDC uses Sunday as the start of the week (https://wwwn.cdc.gov/nndss/document/MMWR_Week_overview.pdf). For example, MSF has varying start dates depending on country in order to better coordinate response.

While there are packages that provide conversion for ISOweeks and epiweeks, these do not provide seamless conversion from dates to epiweeks with non-standard start dates. This package provides a lightweight utility to be able to convert each day.

Note

date2week() will initially convert the input with as.POSIXlt() and use that to calculate the week. If the user supplies character input, it is expected that the input will be of the format yyyy-mm-dd *unless* the user explicitly passes the "format" parameter to as.POSIXlt(). If the input is not in yyyy-mm-dd and the format parameter is not passed, date2week() will result in an error.

Author(s)

Zhian N. Kamvar

See Also

as.Date.aweek(), print.aweek()

Examples

```
# The same set of days will occur in different weeks depending on the start
# date. Here we can define a week before and after today

print(dat <- Sys.Date() + -6:7)

# By default, the weeks are defined as ISO weeks, which start on Monday
print(iso_dat <- date2week(dat))

# If you want lubridate-style numeric-only weeks, you need look no further
# than the "numeric" argument
date2week(dat, 1, numeric = TRUE)

# You can also convert to factor and include all of the missing dates, but
# beware that this may result in a very large factor due to the number of
# levels present
date2week(Sys.Date() + c(0, 10), factor = TRUE)


# The aweek class can be converted back to a date with `as.Date()`
as.Date(iso_dat)

# If you want to show only the first day of the week, you can use the
# `floor_day` argument
as.Date(iso_dat, floor_day = TRUE)

# This also works with `factor`:
as.Date(iso_dat, floor_day = TRUE, factor = TRUE)

# ISO week definition: Monday -- 1
date2week(dat, 1)
date2week(dat, "Monday")

# Tuesday -- 2
date2week(dat, 2)
date2week(dat, "Tuesday")
```

```
# Wednesday -- 3
date2week(dat, 3)
date2week(dat, "W") # you can use valid abbreviations

# Thursday -- 4
date2week(dat, 4)
date2week(dat, "Thursday")

# Friday -- 5
date2week(dat, 5)
date2week(dat, "Friday")

# Saturday -- 6
date2week(dat, 6)
date2week(dat, "Saturday")

# Epiweek definition: Sunday -- 7
date2week(dat, 7)
date2week(dat, "Sunday")
```

`print.aweek` *The aweek class*

Description

The method `c.aweek()` will always return an aweek object with the same `week_start` attribute as the first object in the list. If the first object is also a factor, then the output will be re-leveled. If week-like characters are presented (e.g. "2019-W10-1"), then these are assumed to have the same `week_start` as the first object.

Usage

```
## S3 method for class 'aweek'
print(x, ...)

## S3 method for class 'aweek'
x[i]

## S3 method for class 'aweek'
c(..., recursive = FALSE, use.names = TRUE)
```

Arguments

`x`	an object of class aweek
`...`	a series of aweek objects (unused in `print.aweek()`)
`i`	index for subsetting an aweek object.
`recursive, use.names`	parameters passed on to `unlist()`

Details

The aweek class is a character or factor in the format YYYY-Www(-d) with a "week_start" attribute containing an integer specifying which day of the ISO 8601 week each week should begin. This documentation shows how to print or subset the object.

Value

an object of class aweek

See Also

date2week(), as.Date.aweek()

Examples

```
d <- as.Date("2018-12-20") + 1:40
w <- date2week(d, week_start = "Sunday")
print(w)

# subsetting acts as normal
w[1:10]

# Combining multiple aweek objects
c(w[1], w[3], w[5])

# differing week_start days will default to the attribute of the first
# aweek object
mon <- date2week(Sys.Date(), week_start = "Monday")
mon
c(w, mon)
c(mon, w)

# combining Dates will be coerced to aweek objects under the same rules
c(w, Sys.Date())

# truncated aweek objects will be un-truncated
w2 <- date2week(d[1:5], week_start = "Monday", floor_day = TRUE)
w2
c(w[1:5], w2)
```

Index

[.aweek *(*print.aweek*)*, 5

as.character.aweek *(*as.POSIXlt.aweek*)*, 2
as.Date.aweek *(*as.POSIXlt.aweek*)*, 2
as.Date.aweek(), *4*, *6*
as.POSIXlt(), *2–4*
as.POSIXlt.aweek, 2
aweek, *2*

c.aweek *(*print.aweek*)*, 5
character, *3*

Date, *3*
date2week, 3
date2week(), *2*, *6*

POSIXt, *3*
print.aweek, 5
print.aweek(), *2*, *4*

unlist(), *5*

week2date *(*date2week*)*, 3