

Package: climatrends (via r-universe)

October 12, 2024

Type Package

Title Climate Variability Indices for Ecological Modelling

Version 1.0

URL <https://agrdatasci.github.io/climatrends/>

BugReports <https://github.com/agrdatasci/climatrends/issues>

Description Supports analysis of trends in climate change, ecological and crop modelling.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Depends R (>= 3.5.0), methods

Imports stats, nasapower

Suggests ag5Tools, chirps, knitr, rmarkdown, sf, testthat

Language en-GB

RoxygenNote 7.2.1

VignetteBuilder knitr

Repository <https://agrdatasci.r-universe.dev>

RemoteUrl <https://github.com/agrdatasci/climatrends>

RemoteRef HEAD

RemoteSha 990e692fdc2f684f0a2359f4f5e593eb97d964f3

Contents

climatrends	2
clima_data	2
crop_sensitive	3
cumdrought	5
cumrain	6
ETo	7

GDD	9
get_timeseries	11
late_frost	13
rainfall	16
temperature	18
Index	22

climatrends	<i>Climate Variability Indices for Ecological Modelling</i>
-------------	---

Description

Supports analysis of trends in climate change, ecological and crop modelling.

Author(s)

Kauê de Sousa and Jacob van Etten and Svein Ø. Solberg

See Also

Useful links:

- Development repository: <https://github.com/agrdatasci/climatrends>
- Static documentation: <https://agrdatasci.github.io/climatrends/>
- Report bugs: <https://github.com/agrdatasci/climatrends/issues>

clima_data	<i>Example of input data using local data</i>
------------	---

Description

Input example from disk data. See details

Format

an array with two layers (temp_dat) a matrix (rain_dat), a data.frame (innlandet), a sf object (lon-latsf), a list ("commonbean")

Details

commonbean: a list with three data sets 1) on-farm variety performance data, 2) chirps data, and 3) modis data.

rain_dat: matrix with precipitation from CHIRPS. An excerpt to represent an example of the input data in rainfall() or

get_timeseries() when a matrix is provided.

rain_dat: matrix with precipitation from CHIRPS. An excerpt to represent an example of the input data in rainfall() or get_timeseries() when a matrix is provided.

innlandet: a data.frame with maximum and minimum temperature for a random point in the Innlandet county in Norway, spanning from "2019-01-01" to "2019-07-01"

lonlatsf: a 'sf' object with 'POINT' geometry with five random points around the municipality of Sinop, Brazil

temp_dat and rain_dat, rows represents the coordinates for the given lonlat provided in raster::extract() and columns represents the dates from the observed precipitation/temperature.

Source

Funk, C. et al. (2015). The climate hazards infrared precipitation with stations—a new environmental record for monitoring extremes. *Scientific Data*, 2, 150066. doi:10.1038/sdata.2015.66

van Etten, J. et. al. (2019). PNAS 116(10) 4194-4199 doi:10.1073/pnas.1813720116

Wan Z, Hook S, Hulley G (2015) MYD11A1 MODIS/Aqua Land Surface Temperature/Emissivity 8-Day L3 Global 1km SIN Grid V006 doi:10.5067/MODIS/MYD11A2.006.

crop_sensitive	<i>Crop sensitive indices</i>
----------------	-------------------------------

Description

Compute crop sensitive indices. These indices are designed to capture the changes in temperature extremes during key phenological stages (e.g. anthesis), but can also be applied to other phenological stages.

Usage

```
crop_sensitive(object, ...)

## Default S3 method:
crop_sensitive(object, tmin, ...)

## S3 method for class 'data.frame'
crop_sensitive(object, day.one, ...)

## S3 method for class 'array'
crop_sensitive(object, day.one, ...)
```

```
## S3 method for class 'sf'
crop_sensitive(object, day.one, ..., as.sf = TRUE)
```

Arguments

object	a numeric vector with the maximum temperature, or a data.frame with geographical coordinates (lonlat), or an object of class sf with geometry 'POINT' or 'POLYGON', or an array with two dimensions containing the maximum and minimum temperature, in that order. See details
...	additional arguments passed to methods. See details
tmin	a numeric vector with the minimum temperature
day.one	a vector of class Date or any other object that can be coerced to Date (e.g. integer, character YYYY-MM-DD) for the starting day to capture the climate data
as.sf	logical, to return an object of class 'sf'

Details

The function uses pre-defined threshold to compute the indices. For hts_mean (32, 35, 38 Celsius), for hts_max (36, 39, 42 Celsius), for hse (31 Celsius), for cdi_mean (22, 23, 24 Celsius), for cdi_max (27, 28, 29 Celsius) and for lethal (43, 46, 49 Celsius).

Additional arguments:

The thresholds can be adjusted using the arguments hts_mean.threshold, hts_max.threshold, hse.threshold, cdi_mean.threshold, cdi_max.threshold and lethal.threshold which are a numeric (or vector of numeric)

last.day: an object (optional to span) of class Date or any other object that can be coerced to Date (e.g. integer, character YYYY-MM-DD) for the last day of the time series. For data.frame, array and sf methods

span: an integer (optional to last.day) or a vector with integers (optional if last.day is given) for the length of the time series to be captured. For data.frame, array and sf methods

Value

A dataframe with crop sensitive indices with n columns depending on the number of thresholds passed to each index:

hts_mean	high temperature stress using daily MEAN temperature, and given as percentage number of days a certain threshold is exceeded
hts_max	high temperature stress using daily MAX temperature, and given as percentage number of days a certain threshold is exceeded
hse	heat stress event, and given as percentage number of days a a certain threshold is exceeded for at least two consecutive days
hse_ms	heat stress event, and given the maximum number of days a certain threshold is exceeded for at least two consecutive days

cdi_mean	crop duration index using daily MEAN temperature, and given as $\max(\text{Tmean} - \text{threshold}, 0)$
cdi_max	crop duration index using daily MAX temperature, and given as $\max(\text{Tmax} - \text{threshold}, 0)$
lethal	lethal temperatures, defined as percentage of days during the timeseries where daily MEAN temperature exceeds a given threshold

References

- Challinor et al. (2016). Nature Climate Change 6(10):6954-958 [doi:10.1038/nclimate3061](https://doi.org/10.1038/nclimate3061)
 Trnka et al. (2014). Nature Climate Change 4(7):637-43. [doi:10.1038/nclimate2242](https://doi.org/10.1038/nclimate2242)

See Also

Other temperature functions: [ETo\(\)](#), [GDD\(\)](#), [temperature\(\)](#)

Examples

```
# the default method
set.seed(78)
tmax <- runif(50, 37, 47)
set.seed(79)
tmin <- runif(50, 31, 34)

crop_sensitive(tmax, tmin)

#####

# the array method
data("temp_dat", package = "climatrends")

# use the default thresholds
crop_sensitive(temp_dat,
              day.one = "2013-10-27",
              last.day = "2013-11-04")

# or change the thresholds based on the crop physiology
crop_sensitive(temp_dat,
              day.one = "2013-10-27",
              last.day = "2013-11-04",
              hts_mean.threshold = c(24),
              hts_max.threshold = c(31, 33))
```

cumdrought

Cumulative sum of dry days

Description

Returns a vector with the cumulative sum of the maximum length of dry spell (MLDS)

Usage

```
cumdrought(x)
```

Arguments

```
x          a numeric vector
```

Value

a vector with the cumulative sum of MLDS

Examples

```
rain <- c(0,0.2,0.4,0.1,0.4,5.1,1.5,1.6,0.1,0,7,6,4,6,0.1,1.2,0.5,0)
cumdrought(rain)
```

cumrain

Cumulative sum of rainfall days

Description

Returns a vector with the cumulative sum of the maximum length of wet spell (MLWS)

Usage

```
cumrain(x)
```

Arguments

```
x          a numeric vector
```

Value

a vector with the cumulative sum of MLWS

Examples

```
# Example 1
rain <- c(0,0.2,1.4,6.1,1.4,5.1,1.5,1.6,0.1,0,7,6,4,6,1.1,1.2,1.5,0)
cumrain(rain)

# should return this vector
# raincum <- c(0,0,1,2,3,4,5,6,6,6,6,6,6,6,6,6,7,7)

# Example 2
```

```
rain2 <- c(1,0,1,1,0,0,1,1,1,0,0,1,1)
cumrain(rain2)

# should return this
# raincum2 <- c(1,1,1,2,2,2,2,2,3,3,3,3,3)
```

ETo

Reference evapotranspiration

Description

Reference evapotranspiration using the Blaney-Criddle method. This is general theoretical method used when no measured data on pan evaporation is available locally.

Usage

```
ETo(object, ..., Kc = 1)

## Default S3 method:
ETo(object, tmin, ..., Kc = 1, lat = NULL, month = NULL)

## S3 method for class 'data.frame'
ETo(object, day.one, ..., Kc = 1)

## S3 method for class 'array'
ETo(object, day.one, ..., Kc = 1, lat = NULL, p = 0.27)

## S3 method for class 'sf'
ETo(object, day.one, ..., Kc = 1, as.sf = TRUE)
```

Arguments

object	a numeric vector with the maximum temperature, or a data.frame with geographical coordinates (lonlat), or an object of class sf with geometry 'POINT' or 'POLYGON', or an array with two dimensions containing the maximum and minimum temperature, in that order. See details
...	additional arguments passed to methods. See details
Kc	a numeric value for the crop factor for water requirement
tmin	a numeric vector with the minimum temperature
lat	a vector for the latitude (in Decimal degrees), used to compute mean daily percentage of annual daytime hours based on the latitude and month. This is extracted automatically in the sf method. See details
month	an integer for the reference month of daylight percentage

<code>day.one</code>	a vector of class <code>Date</code> or any other object that can be coerced to <code>Date</code> (e.g. integer, character YYYY-MM-DD) for the starting day to capture the climate data
<code>p</code>	optional if <code>lat</code> is given, a numeric for the mean daily percentage of annual daytime hours ($p = 0.27$ by default)
<code>as.sf</code>	logical, to return an object of class <code>'sf'</code>

Details

When `lat` is provided, it is combined with the month provided in `day.one` to call for the system data `daylight` to find the correct value for `p` which represents the daily percentage of daytime hours in the given month and latitude. Otherwise `p` is set to 0.27 as default.

The `array` method assumes that `object` contains climate data available in your R section; this requires an array with two dimensions, 1st dimension contains the day temperature and 2nd dimension the night temperature, see `help("temp_dat", package = "climatrends")` for an example on input structure.

The `data.frame` method and the `sf` method assumes that the climate data will be fetched from a remote (cloud) source that be adjusted using the argument `data.from`.

Additional arguments:

`last.day`: an object (optional to `span`) of class `Date` or any other object that can be coerced to `Date` (e.g. integer, character YYYY-MM-DD) for the last day of the time series

`span`: an integer (optional to `last.day`) or a vector with integers (optional if `last.day` is given) for the length of the time series to be captured

`data.from`: character for the source of climate data. Current remote data is: `'nasapower'`

`pars`: character vector for the temperature data to be fetched. If `data.from` is `'nasapower'`. The temperature can be adjusted to 2 m, the default, `c("T2M_MAX", "T2M_MIN")` or 10 m `c("T10M_MAX", "T10M_MIN")`

`days.before`: optional, an integer for the number of days before `day.one` to be included in the timespan.

Value

The evapotranspiration in mm/day

References

Brouwer C. & Heibloem M. (1986). Irrigation water management: Irrigation water needs. Food and Agriculture Organization of The United Nations, Rome, Italy. <https://www.fao.org/3/S2022E/s2022e00.htm>

See Also

Other temperature functions: [GDD\(\)](#), [crop_sensitive\(\)](#), [temperature\(\)](#)

Examples

```

# the default method
set.seed(78)
tmax <- runif(50, 37, 47)
set.seed(79)
tmin <- runif(50, 31, 34)

ETo(tmax, tmin, lat = 22, month = 10)

#####

# the array method
data("temp_dat", package = "climatrends")

ETo(temp_dat,
     day.one = "2013-10-28",
     span = 10,
     Kc = 0.92)

```

GDD

Growing degree-days

Description

This a heuristic tool in phenology that measures heat accumulation and is used to predict plant and animal development rates. Growing degree-days are calculated by taking the integral of warmth above a base temperature.

Usage

```

GDD(object, ..., tbase = 10)

## Default S3 method:
GDD(object, tmin, ..., tbase = 10)

## S3 method for class 'data.frame'
GDD(object, day.one, ..., tbase = 10)

## S3 method for class 'array'
GDD(object, day.one, ..., tbase = 10)

## S3 method for class 'sf'
GDD(object, day.one, ..., tbase = 10, as.sf = TRUE)

```

Arguments

<code>object</code>	a numeric vector with the maximum temperature, or a <code>data.frame</code> with geographical coordinates (<code>lonlat</code>), or an object of class <code>sf</code> with geometry 'POINT' or 'POLYGON', or an array with two dimensions containing the maximum and minimum temperature, in that order. See details
<code>...</code>	additional arguments passed to methods. See details
<code>tbase</code>	an integer for the minimum temperature for growth
<code>tmin</code>	a numeric vector with the minimum temperature
<code>day.one</code>	a vector of class <code>Date</code> or any other object that can be coerced to <code>Date</code> (e.g. integer, character YYYY-MM-DD) for the starting day to capture the climate data
<code>as.sf</code>	logical, to return an object of class 'sf'

Details

Additional arguments:

`equation` character to specify the equation to be used, one of "default", "a", "b" or "c". See Equations below

`tbase_max` optional, the maximum `tbase` temperature, required if `equation = "c"`

`return.as` character (one of, the default, "acc" or "daily", "ndays") to select if the function returns the accumulated gdd, or the daily values of gdd across the series, or the number of days required to reach a certain number of degree.days

`degree.days` an integer for the accumulated degree-days required by the organism. Optional if `return.as = "daily"` or `return.as = "acc"`

`last.day`: an object (optional to `span`) of class `Date` or any other object that can be coerced to `Date` (e.g. integer, character YYYY-MM-DD) for the last day of the time series. For `data.frame`, `array` and `sf` methods

`span`: an integer (optional to `last.day`) or a vector with integers (optional if `last.day` is given) for the length of the time series to be captured. For `data.frame`, `array` and `sf` methods

S3 Methods:

The `array` method assumes that `object` contains climate data available in your R section; this requires an array with two dimensions, 1st dimension contains the day temperature and 2nd dimension the night temperature, see `help("temp_dat", package = "climatrends")` for an example on input structure.

The `data.frame` and `sf` methods assumes that the climate data will be fetched from a remote (cloud) source that be adjusted using the argument `data.from`

Equations:

"default": $GDD = ((tmax + tmin) / 2) - tbase$

"a": adjust $tmean = tbase$ if $tmean < tbase$

"b": adjust $tmin = tbase$ if $tmin < tbase$, adjust $tmax = tbase$ if $tmax < tbase$

"c": adjust $tmin = tbase$ if $tmin < tbase$, adjust $tmax = tbase_max$ if $tmax < tbase_max$

Value

The number of days to reach the accumulated *degree.days* or the daily degree-days as defined with the argument *return.as*

References

- Prentice I. C., et al. (1992) *Journal of Biogeography*, 19(2), 117.
 Baskerville, G., & Emin, P. (1969). *Ecology*, 50(3), 514-517. [doi:10.2307/1933912](https://doi.org/10.2307/1933912)

See Also

Other temperature functions: [ETo\(\)](#), [crop_sensitive\(\)](#), [temperature\(\)](#)
 Other GDD functions: [late_frost\(\)](#)

Examples

```
data("innlandet", package = "climatrends")

# use the default equation
GDD(innlandet$tmax, innlandet$tmin, tbase = 2)

# set the equation "b", which is a better option for this case
# tmin = tbase if tmin < tbase
# tmax = tbase if tmax < tbase
GDD(innlandet$tmax, innlandet$tmin, tbase = 2, equation = "b")

#####

# return as the number of days required to reach a certain accumulated GDD
# use equation "c", which adjusts tmax base on a tbase_max
data("temp_dat", package = "climatrends")

GDD(temp_dat,
     day.one = "2013-10-27",
     degree.days = 90,
     return.as = "ndays",
     tbase_max = 32,
     equation = "c")
```

Description

General functions and methods to concatenate climate data across a time series

Usage

```

get_timeseries(object, day.one, ...)

## Default S3 method:
get_timeseries(
  object,
  day.one,
  span = NULL,
  last.day = NULL,
  as.matrix = FALSE,
  data.from = "nasapower",
  ...
)

## S3 method for class 'matrix'
get_timeseries(object, day.one, span = NULL, last.day = NULL, ...)

## S3 method for class 'array'
get_timeseries(object, day.one, span = NULL, last.day = NULL, ...)

```

Arguments

object	a data.frame (or any other object that can be coerced to data.frame) with geographical coordinates (lonlat), or an object of class sf with geometry 'POINT' or 'POLYGON', or a named matrix with climate data, or an array with two dimensions for max and min temperature. See details.
day.one	a vector of class Date or any other object that can be coerced to Date (e.g. integer, character YYYY-MM-DD) for the starting day to capture the climate data
...	additional arguments passed to methods. See details.
span	an integer or a vector with integers (optional if last.day is given) for the length of the time series to be captured
last.day	optional to span, an object of class Date or any other object that can be coerced to Date (e.g. integer, character YYYY-MM-DD) for the last day of the time series
as.matrix	logical, optional, to return a matrix or array instead of a data.frame
data.from	character, for the source of climate data. See details.

Details

The default method and the sf method assumes that the climate data will be fetched from an remote (cloud) *data.from*.

The matrix method assumes that the climate data was previously handled and will be inputted in the format of a named matrix. See help("modis", "climatrends") for examples.

Available remote sources to pass *data.from*: "nasapower"

Additional arguments:

`pars`: character vector of solar, meteorological or climatology parameters to download. See `help("parameters", "nasapower")` when `data.from = "nasapower"`.

`days.before`: an integer for the number of days before `day.one` to be included in the timespan.

Value

A list with class `clima_ls` with `data.frame(s)` with the class `clima_df`

Examples

```
# Using local sources
# an array with temperature data
data("temp_dat", package = "climatrends")

set.seed(9271)
span <- as.integer(runif(10, 6, 15))

get_timeseries(temp_dat, "2013-10-28", span = span)

# matrix with precipitation data
data("rain_dat", package = "climatrends")

get_timeseries(rain_dat, "2013-10-28", span = span)

# data can be returned as matrix
library("sf")
# Fetch data from NASA POWER using 'sf' method
data("lonlatsf", package = "climatrends")

g <- get_timeseries(object = lonlatsf,
                    day.one = "2018-05-16",
                    last.day = "2018-05-30",
                    pars = c("PRECTOTCORR", "T2M"),
                    as.matrix = TRUE)
```

late_frost

Late spring frost

Description

Compute late spring frost, which is a freezing event occurring after a substantial accumulation of warmth

Usage

```
late_frost(object, ..., tbase = 4, tfrost = -2)

## Default S3 method:
late_frost(object, tmin, ..., tbase = 4, tfrost = -2)

## S3 method for class 'data.frame'
late_frost(object, day.one, ..., tbase = 4, tfrost = -2)

## S3 method for class 'array'
late_frost(object, day.one, ..., tbase = 4, tfrost = -2)

## S3 method for class 'sf'
late_frost(object, day.one, ..., tbase = 4, tfrost = -2)
```

Arguments

object	a numeric vector with the maximum temperature, or a data.frame with geographical coordinates (lonlat), or an object of class sf with geometry 'POINT' or 'POLYGON', or an array with two dimensions containing the maximum and minimum temperature, in that order. See details
...	additional arguments passed to methods. See details
tbase	an integer for the minimum temperature for growth
tfrost	an integer for the freezing threshold
tmin	a numeric vector with the minimum temperature
day.one	a vector of class Date or any other object that can be coerced to Date (e.g. integer, character YYYY-MM-DD) for the starting day to capture the climate data

Details

Additional arguments:

equation: character to specify the equation to be used, "b" is set by default. See GDD()

dates: a character (or Date or numeric) vector for the dates of tmax and tmin in the default method

last.day: an object (optional to *span*) of class Date or any other object that can be coerced to Date (e.g. integer, character YYYY-MM-DD) for the last day of the time series

span: an integer (optional to *last.day*) or a vector with integers (optional if *last.day* is given) for the length of the time series to be captured

Value

A data.frame with the late frost events

id	the id generated using the indices for the rows in <i>object</i>
date	the first day of the event

gdd	the growing degree-days accumulated during the event
event	a factor for the observed event, frost, latent (where there is no frost event, but also there is no GDD), and warming (where GDD is accumulated)
duration	the number of days the event spanned

References

Trnka et al. (2014). Nature Climate Change 4(7):637–43. [doi:10.1038/nclimate2242](https://doi.org/10.1038/nclimate2242)

Zohner et al. (2020). PNAS. [doi:10.1073/pnas.1920816117](https://doi.org/10.1073/pnas.1920816117)

See Also

Other GDD functions: [GDD\(\)](#)

Examples

```
# default method
data("innlandet", package = "climatrends")

# equation b is set by default
# where tmin and tmax are adjusted if below tbase
late_frost(innlandet$tmax,
           innlandet$tmin,
           dates = innlandet$date,
           tbase = 2,
           tfrost = -2)

# slightly different series if equation a is used
late_frost(innlandet$tmax,
           innlandet$tmin,
           dates = innlandet$date,
           tbase = 2,
           tfrost = -2,
           equation = "a")

#####

# demo of the array method but no frost event is returned
# because the data comes from the tropics
data("temp_dat", package = "climatrends")

late_frost(temp_dat, day.one = "2013-10-27")
```

rainfall	<i>Rainfall indices</i>
----------	-------------------------

Description

Methods to compute rainfall indices over a time series

Usage

```
rainfall(object, ...)

## Default S3 method:
rainfall(object, ..., timeseries = FALSE)

## S3 method for class 'data.frame'
rainfall(object, day.one, span = NULL, ..., timeseries = FALSE)

## S3 method for class 'matrix'
rainfall(object, day.one, span = NULL, ..., timeseries = FALSE)

## S3 method for class 'sf'
rainfall(object, day.one, span = NULL, ..., timeseries = FALSE, as.sf = TRUE)
```

Arguments

object	a numeric vector with precipitation data or a data.frame with geographical coordinates (lonlat), or an object of class sf with geometry 'POINT' or 'POLYGON', or a named matrix with precipitation data. See details.
...	additional arguments passed to methods. See details.
timeseries	logical, FALSE for a single point time series observation or TRUE for a time series based on <i>intervals</i>
day.one	a vector of class Date or any other object that can be coerced to Date (e.g. integer, character YYYY-MM-DD) for the starting day to capture the climate data
span	an integer or a vector with integers (optional if <i>last.day</i> is given) for the length of the time series to be captured
as.sf	logical, to return an object of class 'sf'

Details

#' Additional arguments:

intervals: an integer (no lower than 5), for the days intervals when *timeseries = TRUE*

last.day: optional to *span*, an object of class Date or any other object that can be coerced to Date (e.g. integer, character YYYY-MM-DD) for the last day of the time series

dates: a character (or Date or numeric) vector for the dates of tmax and tmin in the default method

data.from: character for the source of remote data. Current remote source is: 'nasapower'

pars: character vector for the precipitation data to be fetched. If data.from is 'nasapower', the default precipitation parameter is "PRECTOTCORR".

days.before: optional, an integer for the number of days before *day.one* to be included in the timespan.

S3 Methods

The *matrix* method assumes that *object* contains climate data available in your R section; see `help("rain_dat", package = "climatrends")` for an example on input structure.

The *data.frame* and the *sf* methods assumes that the climate data will e fetched from a remote (cloud) source that be adjusted using the argument *data.from*.

When *timeseries* = TRUE, an *id* is created, which is the index for the rownames of the inputted *object*.

Value

A dataframe with rainfall indices:

MLDS	maximum length of consecutive dry day, rain < 1 mm (days)
MLWS	maximum length of consecutive wet days, rain >= 1 mm (days)
R10mm	number of heavy precipitation days 10 >= rain < 20 mm (days)
R20mm	number of very heavy precipitation days rain >= 20 (days)
Rx1day	maximum 1-day precipitation (mm)
Rx5day	maximum 5-day precipitation (mm)
R95p	total precipitation when rain > 95th percentile (mm)
R99p	total precipitation when rain > 99th percentile (mm)
Rtotal	total precipitation (mm) in wet days, rain >= 1 (mm)
SDII	simple daily intensity index, total precipitation divided by the number of wet days (mm/days)

References

Aguilar E., et al. (2005). Journal of Geophysical Research, 110(D23), D23107. [doi:10.1029/2005JD006119](https://doi.org/10.1029/2005JD006119)

Examples

```
# A vector with precipitation data
set.seed(987219)
rain <- runif(50, min = 0, max = 6)

rainfall(rain)

# Return as timeseries with intervals of 7 days
```

```

dates <- 17650:17699
rainfall(rain, dates = dates, timeseries = TRUE, intervals = 7)

#####

# the matrix method
data("rain_dat", package = "climatrends")

rainfall(rain_dat,
         day.one = "2013-10-28",
         span = 12)

```

temperature

Temperature indices

Description

Methods to compute temperature indices over a time series

Usage

```

temperature(object, ...)

## Default S3 method:
temperature(object, tmin, ..., timeseries = FALSE)

## S3 method for class 'data.frame'
temperature(object, day.one, span = NULL, ..., timeseries = FALSE)

## S3 method for class 'array'
temperature(object, day.one, span = NULL, ..., timeseries = FALSE)

## S3 method for class 'sf'
temperature(
  object,
  day.one,
  span = NULL,
  ...,
  timeseries = FALSE,
  as.sf = TRUE
)

```

Arguments

object a numeric vector with the maximum temperature, or a data.frame with geographical coordinates (lonlat), or an object of class sf with geometry 'POINT'

	or 'POLYGON', or an array with two dimensions containing the maximum and minimum temperature, in that order. See details
...	additional arguments passed to methods. See details
tmin	a numeric vector with the minimum temperature
timeseries	logical, FALSE for a single point time series observation or TRUE for a time series based on <i>intervals</i>
day.one	a vector of class Date or any other object that can be coerced to Date (e.g. integer, character YYYY-MM-DD) for the starting day to capture the climate data
span	an integer or a vector with integers (optional if <i>last.day</i> is given) for the length of the time series to be captured
as.sf	logical, to return an object of class 'sf'

Details

Additional arguments:

intervals: an integer (no lower than 5), for the days intervals when *timeseries = TRUE*

last.day: optional to *span*, an object of class Date or any other object that can be coerced to Date (e.g. integer, character YYYY-MM-DD) for the last day of the time series. For *data.frame*, *array* and *sf* methods

dates: a character (or Date or numeric) vector for the dates of *tmax* and *tmin* in the default method

data.from: character for the source of remote data. Current remote source is: 'nasapower'

pars: character vector for the temperature data to be fetched. If *data.from* is 'nasapower', the temperature can be adjusted to 2 m, the default, c("T2M_MAX", "T2M_MIN") or 10 m c("T10M_MAX", "T10M_MIN")

days.before: optional, an integer for the number of days before *day.one* to be included in the timespan.

S3 Methods

The *array* method assumes that *object* contains climate data available in your R section; this requires an array with two dimensions, 1st dimension contains the day temperature and 2nd dimension the night temperature, see help("temp_dat", package = "climatrends") for an example on input structure.

The *data.frame* and the *sf* methods assumes that the climate data will be fetched from a remote (cloud) source that be adjusted using the argument *data.from*.

When *timeseries = TRUE*, an *id* is created, which is the index for the rownames of the inputted *object*.

Value

A dataframe with temperature indices:

maxDT	maximum day temperature (degree Celsius)
minDT	minimum day temperature (degree Celsius)

maxNT	maximum night temperature (degree Celsius)
minNT	minimum night temperature (degree Celsius)
DTR	diurnal temperature range (mean difference between DT and NT (degree Celsius))
SU	summer days, number of days with maximum temperature > 30 (degree Celsius)
TR	tropical nights, number of nights with maximum temperature > 25 (degree Celsius)
CFD	consecutive frosty days, number of days with temperature below 0 degree Celsius
WSDI	maximum warm spell duration, consecutive days with temperature > 90th percentile
CSDI	maximum cold spell duration, consecutive nights with temperature < 10th percentile
T10p	the 10th percentile of night temperature (degree Celsius)
T90p	the 90th percentile of day temperature (degree Celsius)

References

Aguilar E., et al. (2005). Journal of Geophysical Research, 110(D23), D23107. doi:[10.1029/2005JD006119](https://doi.org/10.1029/2005JD006119)

See Also

Other temperature functions: [ETo\(\)](#), [GDD\(\)](#), [crop_sensitive\(\)](#)

Examples

```
# the default method
data("innlandet", package = "climatrends")

# a single temporal observation
temperature(innlandet$tmax, innlandet$tmin)

# return as timeseries with 30-day intervals
temperature(innlandet$tmax,
            innlandet$tmin,
            dates = innlandet$dates,
            timeseries = TRUE,
            intervals = 30)

#####

# array method
data("temp_dat", package = "climatrends")

temperature(temp_dat,
            day.one = "2013-10-28",
            span = 12)
```


Index

* GDD functions

GDD, [9](#)

late_frost, [13](#)

* GET functions

get_timeseries, [11](#)

* datasets

clima_data, [2](#)

* precipitation functions

rainfall, [16](#)

* temperature functions

crop_sensitive, [3](#)

ETo, [7](#)

GDD, [9](#)

temperature, [18](#)

clima_data, [2](#)

climatrends, [2](#)

climatrends-package (climatrends), [2](#)

commonbean (clima_data), [2](#)

crop_sensitive, [3](#), [8](#), [11](#), [20](#)

cumdrought, [5](#)

cumrain, [6](#)

ETo, [5](#), [7](#), [11](#), [20](#)

GDD, [5](#), [8](#), [9](#), [15](#), [20](#)

get_timeseries, [11](#)

innlandet (clima_data), [2](#)

late_frost, [11](#), [13](#)

lonlatsf (clima_data), [2](#)

rain_dat (clima_data), [2](#)

rainfall, [16](#)

temp_dat (clima_data), [2](#)

temperature, [5](#), [8](#), [11](#), [18](#)