Package 'heatwaveR'

January 16, 2019

Version 0.3.6

Date 2019-01-16

Title Detect Heatwaves and Cold-Spells

Description
The different methods of defining and detecting extreme events, known as heatwaves or cold-spells in both air and water temperature data are encompassed within this package. These detection algorithms may be used on non-temperature data as well however, this is not catered for explicitly here as no use of this technique in the literature currently exists.

Type Package

Maintainer Robert W. Schlegel <robwschlegel@gmail.com>

License MIT + file LICENSE

Encoding UTF-8

LazyData true

ByteCompile true

RoxygenNote 6.1.1

Depends R (>= 3.0.2), data.table, ggplot2

Suggests tidyverse, ggpubr, testthat, knitr, rmarkdown, covr, rerddap, ncdf4, weathercan

Imports tibble, lubridate, dplyr, stats, utils, zoo, grid, RcppRoll

LinkingTo Rcpp (>= 0.12.16), RcppArmadillo

NeedsCompilation yes

VignetteBuilder knitr

URL https://robwschlegel.github.io/heatwaveR/index.html, https://github.com/robwschlegel/heatwaveR

BugReports https://github.com/robwschlegel/heatwaveR/issues

Author Robert W. Schlegel [aut, cre, ctb] (<https://orcid.org/0000-0002-0705-1287>), Albertus J. Smit [aut, ctb] (<https://orcid.org/0000-0002-3799-6126>)

Repository CRAN

Date/Publication 2019-01-16 20:30:03 UTC

R topics documented:

`block_average` *Calculate yearly means for event metrics.*

Description

Calculate yearly means for event metrics.

Usage

```
block_average(data, x = t, y = temp, report = "full")
```

Arguments

`data`	Accepts the data returned by the `detect_event` function.
`x`	This column is expected to contain a vector of dates as per the specification of `ts2clm`. If a column headed `t` is present in the dataframe, this argument may be ommitted; otherwise, specify the name of the column with dates here.
`y`	This is a column containing the measurement variable. If the column name differs from the default (i.e. `temp`), specify the name here.
`report`	Specify either `full` or `partial`. Selecting `full` causes the report to contain NAs for any years in which no events were detected (except for `count`, which will be zero in those years), while `partial` reports only the years wherein events were detected. The default is `full`.

Details

This function needs to be provided with the full output from the detect_event or exceedance functions. Note that the yearly averages are calculted only for complete years (i.e. years that start/end part-way through the year at the beginning or end of the original time series are removed from the calculations).

This function differs from the python implementation of the function of the same name (i.e., blockAverage, see https://github.com/ecjoliver/marineHeatWaves) in that we only provide the ability to calculate the average (or aggregate) event metrics in 'blocks' of one year, while the python version allows arbitrary (integer) block sizes.

Note that if this function is used on the output of exceedance, all of the metrics (see below) with relThresh in the name will be returned as NA values.

Value

The function will return a data frame of the averaged (or aggregate) metrics. It includes the following:

year	The year over which the metrics were averaged.
count	The number of events per year.
duration	The average duration of events per year [days].
duration_max	The maximum duration of an event in each year [days].
intensity_mean	The average event "mean intensity" in each year [deg. C].
intensity_max	The average event "maximum (peak) intensity" in each year [deg. C].
intensity_max_max	The maximum event "maximum (peak) intensity" in each year [deg. C].
intensity_var	The average event "intensity variability" in each year [deg. C].
intensity_cumulative	The average event "cumulative intensity" in each year [deg. C x days].
rate_onset	Average event onset rate in each year [deg. C / days].
rate_decline	Average event decline rate in each year [deg. C / days].
total_days	Total number of events days in each year [days].
total_icum	Total cumulative intensity over all events in each year [deg. C x days].

intensity_max_relThresh, intensity_mean_relThresh, intensity_var_relThresh, and intensity_cumulative_re are as above except relative to the threshold (e.g., 90th percentile) rather than the seasonal climatology.

intensity_max_abs, intensity_mean_abs, intensity_var_abs, and intensity_cumulative_abs are as above except as absolute magnitudes rather than relative to the seasonal climatology or threshold.

Author(s)

Albertus J. Smit, Eric C. J. Oliver, Robert W. Schlegel

References

Hobday, A.J. et al. (2016), A hierarchical approach to defining marine heatwaves, Progress in Oceanography, 141, pp. 227-238, doi: 10.1016/j.pocean.2015.12.014

Examples

```
ts <- ts2clm(sst_WA, climatologyPeriod = c("1983-01-01", "2012-12-31"))
res <- detect_event(ts)
out <- block_average(res)
summary(glm(count ~ year, out, family = "poisson"))

library(ggplot2)

ggplot(data = out, aes(x = year, y = count)) +
  geom_point(colour = "salmon") +
  geom_line() +
  labs(x = NULL, y = "Number of events")
```

`category` *Calculate the categories of events.*

Description

Calculates the categories of a series of events as produced by `detect_event` in accordance with the naming scheme proposed in Hobday et al. (2018).

Usage

```
category(data, y = temp, S = TRUE, name = "Event",
  climatology = FALSE)
```

Arguments

`data`	The function receives the full (list) output from the `detect_event` function.
`y`	The column containing the measurement variable. If the column name differs from the default (i.e. `temp`), specify the name here.
`S`	This argument informs the function if the data were collected in the southern hemisphere (TRUE, default) or the northern hemisphere (FALSE) so that it may correctly output the `season` column (see below).
`name`	If a character string (e.g. "Bohai Sea") is provide here it will be used to name the events in the `event_name` column (see below) of the output. If no value is provided the default output is "Event".
`climatology`	The default setting of `FALSE` will tell this function to output only the summary (wide) results for the individual events as seen in Hobday et al. (2018). If set to `TRUE`, this function will rather return a list of two data frames, same as

detect_event. The first dataframe climatology, contains the same information as found in detect_event, but with the addition of the daily intensity and category values. The second dataframe, event, is the summary results that this function produces by default.

Details

An explanation for the categories is as follows:

1. I Moderate-Events that have been detected, but with a maximum intensity that does not double the distance between the seasonal climatology and the threshold value.
2. II Strong-Events with a maximum intensity that doubles the distance from the seasonal climatology and the threshold, but do not triple it.
3. III Severe-Events that triple the aforementioned distance, but do not quadruple it.
4. IV Extreme-Events with a maximum intensity that is four times or greater the aforementioned distance. Scary stuff...

Value

The function will return a tibble with results similar to those seen in Table 2 of Hobday et al. (2018). This provides the information necessary to appraise the extent of the events in the output of detect_event based on the category ranking scale. The category thresholds are calculated based on the difference between the given seasonal climatology and threshold climatology. The four category levels are then the difference multiplied by the category level.

The definitions for the default output columns are as follows:

event_no	The number of the event as determined by detect_event for reference between the outputs.
event_name	The name of the event. Generated from the name value provided and the year of the peak_date (see following) of the event. If no name value is provided the default "Event" is used. As proposed in Hobday et al. (2018), Moderate events are not given a name so as to prevent multiple repeat names within the same year. If two or more events ranked greater than Moderate are reported withiin the same year, they will be differentiated with the addition of a trailing letter (e.g. Event 2001 a, Event 2001 b). (still in development)
peak_date	The date (day) on which the maximum intensity of the event was recorded.
category	The maximum category threshold reached/exceeded by the event.
i_max	The maximum intensity of the event above the threshold value.
duration	The total duration (days) of the event. Note that this includes any possible days when the measurement value y) may have dropped below the threshold value. Therefore, the proportion of the event duration (days) spent above certain thresholds may not add up to 100% (see following four items).
p_moderate	The proportion of the total duration (days) spent at or above the first threshold, but below any further thresholds.
p_strong	The proportion of the total duration (days) spent at or above the second threshold, but below any further thresholds.

p_severe	The proportion of the total duration (days) spent at or above the third threshold, but below the fourth threshold.
p_extreme	The proportion of the total duration (days) spent at or above the fourth and final threshold. There is currently no recorded event that has exceeded a hypothetical fifth threshold so none is calculated... yet..
season	The season(S) during which the event occurred. If the event occurred across two seasons this will be displayed as "Winter/Spring". Across three seasons as "Winter-Summer". Events lasting across four or more seasons are listed as "Year-round". December (June) is used here as the start of Austral (Boreal) summer.

If `climatology = TRUE`, this function will output a list of two dataframes. The first dataframe, `climatology`, will contain only the following columns:

t	The column containing the daily date values.
event_no	The numeric event number label.
intensity	The total exceedance (default is degrees C) above the 90th percentile threshold.
category	The category classification per day.

The second dataframe, `event`, contains the default output of this function, as detailed above.

Author(s)

Robert W. Schlegel

References

Hobday et al. (2018). Categorizing and Naming Marine Heatwaves. Oceanography 31(2).

Examples

```
res_WA <- detect_event(ts2clm(sst_WA,
                       climatologyPeriod = c("1983-01-01", "2012-12-31")))
# Note that the name argument expects a character vector
cat_WA <- category(res_WA, name = "WA")
tail(cat_WA)

# If the data were collected in the northern hemisphere
# we must let the funciton know this, as seen below
res_Med <- detect_event(ts2clm(sst_Med,
                        climatologyPeriod = c("1983-01-01", "2012-12-31")))
cat_Med <- category(res_Med, S = FALSE, name = "Med")
tail(cat_Med)

# One may also choose to have this function output the daily
# category classifications as well by setting: climatology = TRUE
cat_WA_daily <- category(res_WA, name = "WA", climatology = TRUE)
head(cat_WA_daily$climatology)

# Note that this will not return the complete time series, only the
```

```
# days during which events were detected.
# This was done to reduce the size of the output for those working
# with gridded data.
# Should one want a complete time series, the daily category results
# may simply be left_join() with the detect_event() results
cat_WA_ts <- dplyr::left_join(res_WA$climatology,
                              cat_WA_daily$climatology)
head(cat_WA_ts)
```

`clim_calc` *Calculate seasonal and threshold climatologies as well as the variance.*

Description

An internal function that helps to create the climatologies that are then output with `ts2clm`.

Usage

```
clim_calc(data, windowHalfWidth, pctile)
```

Arguments

`data`	The data given to this function during the calculations performed by `ts2clm`.
`windowHalfWidth`	The width of the smoothing window to be applied. This width is doubled and centred around the point that the smoothing occurs. Default = 5, which makes an overall window size of 11.
`pctile`	Threshold percentile (%) for detection of events (MHWs). Default is `90`th percentile.

Value

The function returns the calculated climatologies.

Author(s)

Smit, A. J.

clim_spread *Spead a time series wide to allow for a climatology to be calculated.*

Description

An internal function that helps to create a wide time series that will then be used by `clim_calc` within `ts2clm` to produce a climatology as desired by the user.

Usage

```
clim_spread(data, clim_start, clim_end, windowHalfWidth)
```

Arguments

`data`	The data given to this function during the calculations performed by `ts2clm`.
`clim_start`	The first day of the time series to use when spreading.
`clim_end`	The last day of the time series to use when spreading.
`windowHalfWidth`	The width of the smoothing window to be applied. This width is doubled and centred around the point that the smoothing occurs. Default = 5, which makes an overall window size of 11.

Value

The function returns the data (a matrix) in a wide format.

Author(s)

Smit, A. J.

detect_event *Detect heatwaves and cold-spells.*

Description

Applies the Hobday et al. (2016) marine heat wave definition to an input time series of a given value (usually, but not necessarily limited to, temperature) along with a daily date vector and pre-calculated seasonal and threshold climatologies, which may either be created with `ts2clm` or some other means.

Usage

```
detect_event(data, x = t, y = temp, seasClim = seas,
  threshClim = thresh, threshClim2 = NA, minDuration = 5,
  minDuration2 = minDuration, joinAcrossGaps = TRUE, maxGap = 2,
  maxGap2 = maxGap, coldSpells = FALSE, protoEvents = FALSE)
```

Arguments

data	A data frame with at least four columns. In the default setting (i.e. ommitting the arguments x, y, seas, and thresh; see immediately below), the data set is expected to have the headers t, temp, seas, and thresh. The t column is a vector of dates of class Date, temp is the measured variable (by default it is assumed to be temperature), seas is the seasonal cycle daily climatology (366 days), and thresh is the seasonal cycle daily threshold above which events may be detected. Data of the appropriate format are created by the function ts2clm, but your own data can be supplied if they meet the criteria specified by ts2clm If the column names of data match those outlined here, the following four arguments may be ignored.
x	This column is expected to contain a vector of dates as per the specification of ts2clm. If a column headed t is present in the dataframe, this argument may be ommitted; otherwise, specify the name of the column with dates here.
y	This is a column containing the measurement variable. If the column name differs from the default (i.e. temp), specify the name here.
seasClim	This function will assume that the seasonal climatology column is called seas as this matches the output of ts2clm. If the column name for the seasonal climatology is different, provide that here.
threshClim	The threshold climatology column should be called thresh. If it is not, provide the name of the threshold column here.
threshClim2	If one wishes to provide a second climatology threshold filter for the more rigorous detection of events, a vector or column conataining logical values (i.e. TRUE FALSE) should be provided here. By default this argument is ignored. It's primary purpose is to allow for the inclusion of tMin and tMax thresholds.
minDuration	The minimum duration for acceptance of detected events. The default is 5 days.
minDuration2	The minimum duration for acceptance of events after filtering by threshClim and threshClim. By default minDuration2 = minDuration and is ignored if threshClim2 has not been specified.
joinAcrossGaps	Boolean switch indicating whether to join events which occur before/after a short gap as specified by maxGap. The default is TRUE.
maxGap	The maximum length of gap allowed for the joining of MHWs. The default is 2 days.
maxGap2	The maximum gap length after applying both thresholds. By default maxGap2 = maxGap and is ignored if threshClim2 has not been specified.
coldSpells	Boolean specifying if the code should detect cold events instead of warm events. The default is FALSE. Please note that the climatological thresholds for coldspells are considered to be the inverse of those for MHWs. For example, the default setting for the detection of MHWs is pctile = 90, as seen in ts2clm. Should one want to use detect_event for MCSs, this threshold would best be generated in ts2clm by setting pctile = 10 (see example below). Any value may be used, but this is the setting used for the calculation of MCSs in Schlegel et al. (2017a).

protoEvents Boolean specifying whether the full time series must be returned as a long table, together with columns indicating whether or not the threshold criterion (threshCriterion) and duration criterion (durationCriterion) have been exceeded, a column showing if a heatwave is present (i.e. both threshCriterion and durationCriterion TRUE), and a sequential number uniquely identifying the detected event. In this case, the heatwave metrics will not be reported. The default is FALSE.

Details

1. This function assumes that the input time series consists of continuous daily values with few missing values. Time ranges which start and end part-way through the calendar year are supported. The accompanying function ts2clm aids in the preparation of a time series that is suitable for use with detect_event, although this may also be accomplished 'by hand' as long as the criteria are met as discussed in the documentation to ts2clm.
2. The calculation of onset and decline rates assumes that the events started a half-day before the start day and ended a half-day after the end-day. This is consistent with the duration definition as implemented, which assumes duration = end day - start day + 1. An event that is already present at the beginning of a time series, or an event that is still present at the end of a time series, will report the rate of onset or the rate of decline as NA, as it is impossible to know what the temperature half a day before or after the start or end of the event is.
3. For the purposes of event detection, any missing temperature values not interpolated over (through optional maxPadLength in ts2clm) will be set equal to the seasonal climatology. This means they will trigger the end/start of any adjacent temperature values which satisfy the event definition criteria.
4. If the code is used to detect cold events (coldSpells = TRUE), then it works just as for heat waves except that events are detected as deviations below the (100 - pctile)th percentile (e.g., the 10th instead of 90th) for at least 5 days. Intensities are reported as negative values and represent the temperature anomaly below climatology.

The original Python algorithm was written by Eric Oliver, Institute for Marine and Antarctic Studies, University of Tasmania, Feb 2015, and is documented by Hobday et al. (2016). The marine cold spell option was implemented in version 0.13 (21 Nov 2015) of the Python module as a result of our preparation of Schlegel et al. (2017), wherein the cold events receive a brief overview.

Value

The function will return a list of two tibbles (see the tidyverse), climatology and event, which are, surprisingly, the climatology and event results, respectively. The climatology contains the full time series of daily temperatures, as well as the the seasonal climatology, the threshold and various aspects of the events that were detected. The software was designed for detecting extreme thermal events, and the units specified below reflect that intended purpose. However, various other kinds of extreme events may be detected according to the specifications, and if that is the case, the appropriate units need to be determined by the user.

The climatology results will contain the same column produced by ts2clm as well as the following:

threshCriterion
Boolean indicating if temp exceeds thresh.

`durationCriterion` Boolean indicating whether periods of consecutive `threshCriterion` are >= `min_duration`.

`event` Boolean indicating if all criteria that define an extreme event are met.

`event_no` A sequential number indicating the ID and order of occurence of the events.

The `event` results are summarised using a range of event metrics:

`event_no` A sequential number indicating the ID and order of the events.

`index_start` Start index of event.

`index_end` End index of event.

`duration` Duration of event [days].

`date_start` Start date of event [date].

`date_end` End date of event [date].

`date_peak` Date of event peak [date].

`intensity_mean` Mean intensity [deg. C].

`intensity_max` Maximum (peak) intensity [deg. C].

`intensity_var` Intensity variability (standard deviation) [deg. C].

`intensity_cumulative` Cumulative intensity [deg. C x days].

`rate_onset` Onset rate of event [deg. C / day].

`rate_decline` Decline rate of event [deg. C / day].

`intensity_max_relThresh`, `intensity_mean_relThresh`, `intensity_var_relThresh`, and `intensity_cumulative_re` are as above except relative to the threshold (e.g., 90th percentile) rather than the seasonal climatology.

`intensity_max_abs`, `intensity_mean_abs`, `intensity_var_abs`, and `intensity_cumulative_abs` are as above except as absolute magnitudes rather than relative to the seasonal climatology or threshold.

Note that `rate_onset` and `rate_decline` will return `NA` when the event begins/ends on the first/last day of the time series. This may be particularly evident when the function is applied to large gridded data sets. Although the other metrics do not contain any errors and provide sensible values, please take this into account in its interpretation.

Author(s)

Albertus J. Smit, Robert W. Schlegel, Eric C. J. Oliver

References

Hobday, A.J. et al. (2016). A hierarchical approach to defining marine heatwaves, Progress in Oceanography, 141, pp. 227-238, doi:10.1016/j.pocean.2015.12.014

Schlegel, R. W., Oliver, C. J., Wernberg, T. W., Smit, A. J. (2017). Nearshore and offshore co-occurrences of marine heatwaves and cold-spells. Progress in Oceanography, 151, pp. 189-205, doi:10.1016/j.pocean.2017.01.004

Examples

```
res_clim <- ts2clm(sst_WA, climatologyPeriod = c("1983-01-01", "2012-12-31"))
out <- detect_event(res_clim)
# show a portion of the climatology:
out$climatology[1:10, ]
# show some of the heat waves:
out$event[1:5, 1:10]

# Or if one wants to calculate MCSs
res_clim <- ts2clm(sst_WA, climatologyPeriod = c("1983-01-01", "2012-12-31"),
                   pctile = 10)
out <- detect_event(res_clim, coldSpells = TRUE)
# show a portion of the climatology:
out$climatology[1:10, ]
# show some of the cold-spells:
out$event[1:5, 1:10]

# It is also possible to give two separate sets of threshold criteria

# To use a second static threshold we first use the exceedance function
thresh_19 <- exceedance(sst_Med, threshold = 19, minDuration = 10, maxGap = 0)$threshold
# Then we use that output when detecting our events
events_19 <- detect_event(ts2clm(sst_Med, climatologyPeriod = c("1982-01-01", "2011-12-31")),
                      threshClim2 = thresh_19$exceedance, minDuration2 = 10, maxGap2 = 0)

# If we want to use two different percentile thresholds we use detect_event
thresh_95 <- detect_event(ts2clm(sst_Med, pctile = 95,
                                 climatologyPeriod = c("1982-01-01", "2011-12-31")),
                          minDuration = 2, maxGap = 0)$climatology
# Then we use that output when detecting our events
events_95 <- detect_event(ts2clm(sst_Med, climatologyPeriod = c("1982-01-01", "2011-12-31")),
                          threshClim2 = thresh_95$event, minDuration2 = 2, maxGap2 = 0)
```

---

event_line *Create a line plot of heatwaves or cold-spells.*

Description

Creates a graph of warm or cold events as per the second row of Figure 3 in Hobday et al. (2016).

Usage

```
event_line(data, x = t, y = temp, min_duration = 5, spread = 150,
  metric = "intensity_cumulative", start_date = NULL,
  end_date = NULL, category = FALSE)
```

Arguments

data	The function receives the full (list) output from the detect_event function.
x	This column is expected to contain a vector of dates as per the specification of make_whole. If a column headed t is present in the dataframe, this argument may be ommitted; otherwise, specify the name of the column with dates here.
y	This is a column containing the measurement variable. If the column name differs from the default (i.e. temp), specify the name here.
min_duration	The minimum duration (days) the event must be for it to qualify as a heatwave or cold-spell.
spread	The number of days leading and trailing the largest event (as per metric) detected within the time period specified by start_date and end_date. The default is 150 days.
metric	This tells the function how to choose the event that should be highlighted as the 'greatest' of the events in the chosen period. One may choose from the following options: intensity_mean, intensity_max, intensity_var,intensity_cumulative, intensity_mean_relThresh, intensity_max_relThresh, intensity_var_relThresh, intensity_cumulative_relThresh, intensity_mean_abs, intensity_max_abs, intensity_var_abs, intensity_cumulative_abs, rate_onset, rate_decline. Partial name matching is currently not supported so please specify the metric name precisely. The default is intensity_cumulative.
start_date	The start date of a period of time within which the largest event (as per metric) is retrieved and plotted. This may not necessarily correspond to the biggest event of the specified metric within the entire time series. To plot the largest event within the whole time series, make sure start_date and end_date straddle this event, or simply leave them both as NULL (default) and event_line will use the entire time series date range.
end_date	The end date of a period of time within which the largest event (as per metric) is retrieved and plotted. See start_date for additional information.
category	A boolean choice of TRUE or FALSE. If set to FALSE (default) event_line() will produce a figure as per the second row of Figure 3 in Hobday et al. (2016). If set to TRUE a figure showing the different categories of the MHWs in the chosen period, highlighted as seen in Figure 3 of Hobday et al. (in review), will be produced. If category = TRUE, metric will be ignored as a different colouring scheme is used.

Value

The function will return a line plot indicating the climatology, threshold and temperature, with the hot or cold events that meet the specifications of Hobday et al. (2016) shaded in as appropriate. The plotting of hot or cold events depends on which option is specified in detect_event. The top event detect during the selected time period will be visible in a brighter colour. This function differs in use from geom_flame in that it creates a stand alone figure. The benefit of this being that one must not have any prior knowledge of ggplot2 to create the figure.

Author(s)

Robert W. Schlegel

References

Hobday, A.J. et al. (2016), A hierarchical approach to defining marine heatwaves, Progress in Oceanography, 141, pp. 227-238, doi: 10.1016/j.pocean.2015.12.014

Examples

```
ts <- ts2clm(sst_WA, climatologyPeriod = c("1983-01-01", "2012-12-31"))
res <- detect_event(ts)

event_line(res, spread = 100, metric = "intensity_cumulative",
start_date = "2010-12-01", end_date = "2011-06-30")

event_line(res, spread = 100, start_date = "2010-12-01",
end_date = "2011-06-30", category = TRUE)
```

---

exceedance *Detect consecutive days in exceedance of a given threshold.*

Description

Detect consecutive days in exceedance of a given threshold.

Usage

```
exceedance(data, x = t, y = temp, threshold, below = FALSE,
  minDuration = 5, joinAcrossGaps = TRUE, maxGap = 2,
  maxPadLength = FALSE)
```

Arguments

data	A data frame with at least the two following columns: a `t` column which is a vector of dates of class `Date`, and a `temp` column, which is the temperature on those given dates. If columns are named differently, their names can be supplied as `x` and `y` (see below). The function will not accurately detect consecutive days of temperatures in exceedance of the `threshold` if missing days of data are not filled in with `NA`. Data of the appropriate format are created by the function `make_whole`, but your own data may be used directly if they meet the given criteria.
x	This column is expected to contain a vector of dates as per the specification of `make_whole`. If a column headed `t` is present in the dataframe, this argument may be ommitted; otherwise, specify the name of the column with dates here.
y	This is a column containing the measurement variable. If the column name differs from the default (i.e. `temp`), specify the name here.
threshold	The static threshold used to determine how many consecutive days are in exceedance of the temperature of interest.

below	Default is FALSE. When set to TRUE, consecutive days of temperature below the threshold variable are calculated. When set to FALSE, consecutive days above the threshold variable are calculated.
minDuration	Minimum duration that temperatures must be in exceedance of the threshold variable. The default is 5 days.
joinAcrossGaps	A TRUE/FALSE statement that indicates whether or not to join consecutive days of temperatures in exceedance of the threshold across a small gap between groups before/after a short gap as specified by maxGap. The default is TRUE.
maxGap	The maximum length of the gap across which to connect consecutive days in exceedance of the threshold when joinAcrossGaps = TRUE.
maxPadLength	Specifies the maximum length of days over which to interpolate (pad) missing data (specified as NA) in the input temperature time series; i.e., any consecutive blocks of NAs with length greater than maxPadLength will be left as NA. Set as an integer. The default is 3 days.

Details

1. This function assumes that the input time series consists of continuous daily temperatures, with few missing values. The accompanying function make_whole aids in the preparation of a time series that is suitable for use with exceedance, although this may also be accomplished 'by hand' as long as the criteria are met as discussed in the documentation to make_whole.
2. Future versions seek to accomodate monthly and annual time series, too.
3. The calculation of onset and decline rates assumes that exceedance of the threshold started a half-day before the start day and ended a half-day after the end-day. This is consistent with the duration definition as implemented, which assumes duration = end day - start day + 1.
4. For the purposes of exceedance detection, any missing temperature values not interpolated over (through optional maxPadLength) will remain as NA. This means they will trigger the end of an exceedance if the adjacent temperature values are in exceedance of the threshold.
5. If the function is used to detect consecutive days of temperature under the given theshold, these temperatures are then taken as being in exceedance below the threshold as there is no antonym in the English language for 'exceedance'.

This function is based largely on the detect_event function found in this package, which was ported from the Python algorithm that was written by Eric Oliver, Institute for Marine and Antarctic Studies, University of Tasmania, Feb 2015, and is documented by Hobday et al. (2016).

Value

The function will return a list of two components. The first being threshold, which shows the daily temperatures and on which specific days the given threshold was exceeded. The second component of the list is exceedance, which shows a medley of statistics for each discrete group of days in exceedance of the given threshold. Note that any additional columns left in the data frame given to this function will be output in the threshold component of the output. For example, if one uses ts2clm to prepare a time series for analysis and leaves in the doy column, this column will appear in the output.

The information shown in the threshold component is:

t	The date of the temperature measurement. This variable may named differently if an alternative name is supplied to the function's x argument.
temp	Temperature on the specified date [deg. C]. This variable may named differently if an alternative name is supplied to the function's y argument.
thresh	The static threshold chosen by the user [deg. C].
thresh_criterion	Boolean indicating if temp exceeds threshold.
duration_criterion	Boolean indicating whether periods of consecutive thresh_criterion are >= minDuration.
exceedance	Boolean indicting if all criteria that define a discrete group in exceedance of the threshold are met.
exceedance_no	A sequential number indicating the ID and order of occurence of exceedances.

The individual exceedances are summarised using the following metrics:

exceedance_no	The same sequential number indicating the ID and order of the exceedance as found in the threshold component of the output list.
index_start	Row number on which exceedance starts.
index_peak	Row number on which exceedance peaks.
index_end	Row number on which exceedance ends.
duration	Duration of exceedance [days].
date_start	Start date of exceedance [date].
date_peak	Date of exceedance peak [date].
date_end	End date of exceedance [date].
intensity_mean	Mean intensity [deg. C].
intensity_max	Maximum (peak) intensity [deg. C].
intensity_var	Intensity standard deviation [deg. C].
intensity_cumulative	Cumulative intensity [deg. C x days].
rate_onset	Onset rate of exceedance [deg. C / day].
rate_decline	Decline rate of exceedance [deg. C / day].

intensity_max_abs, intensity_mean_abs, intensity_var_abs, and intensity_cum_abs are as above except as absolute magnitudes rather than relative to the threshold.

Author(s)

Robert W. Schlegel, Albertus J. Smit

Examples

```
res <- exceedance(sst_WA, threshold = 25)
# show first ten days of daily data:
res$threshold[1:10, ]
# show first five exceedances:
res$exceedance[1:5, ]
```

geom_flame *Create 'flame' polygons.*

Description

This function will create polygons between two lines. If given a temperature and theshold time series, like that produced by detect_event, the output will meet the specifications of Hobday et al. (2016) shown as 'flame polygons.' If one wishes to plot polygons below a given threshold, and not above, switch the values being fed to the y and y2 aesthetics. This function differs in use from event_line in that it must be created as a ggplot 'geom' object. The benefit of this being that one may add additional information to the figure as geom layers to ggplot2 graphs as may be necessary.

Usage

```
geom_flame(mapping = NULL, data = NULL, stat = "identity",
  position = "identity", ..., na.rm = FALSE, show.legend = NA,
  inherit.aes = TRUE)
```

Arguments

mapping	Set of aesthetic mappings created by aes() or aes_(). If specified and inherit.aes = TRUE (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.
data	The data to be displayed in this layer. There are three options: If NULL, the default, the data is inherited from the plot data as specified in the call to ggplot(). A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify() for which variables will be created. A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data.
stat	The statistical transformation to use on the data for this layer, as a string.
position	Position adjustment, either as a string, or the result of a call to a position adjustment function.
...	other arguments passed on to layer. These are often aesthetics, used to set an aesthetic to a fixed value, like color = "red" or size = 3. They may also be parameters to the paired geom/stat.
na.rm	If FALSE (the default), removes missing values with a warning. If TRUE silently removes missing values.
show.legend	Logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes. It can also be a named logical vector to finely select the aesthetics to display.
inherit.aes	If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. borders().

Aesthetics

geom_flame understands the following aesthetics (required aesthetics are in bold):

- x
- y
- y2
- colour
- fill
- size
- alpha
- linetype

Author(s)

Robert W. Schlegel

References

Hobday, A.J. et al. (2016), A hierarchical approach to defining marine heatwaves, Progress in Oceanography, 141, pp. 227-238, doi: 10.1016/j.pocean.2015.12.014

See Also

event_line for a non-ggplot2 based flame function.

Examples

```
ts <- ts2clm(sst_WA, climatologyPeriod = c("1983-01-01", "2012-12-31"))
res <- detect_event(ts)
mhw <- res$clim
mhw <- mhw[10580:10690,]

library(ggplot2)

ggplot(mhw, aes(x = t, y = temp)) +
  geom_flame(aes(y2 = thresh)) +
  geom_text(aes(x = as.Date("2011-02-01"), y = 28,
            label = "That's not a heatwave.\nThis, is a heatwave.")) +
  xlab("Date") + ylab(expression(paste("Temperature [", degree, "C]")))
```

geom_lolli *Visualise a timeline of several event metrics as 'lollipops'.*

Description

The function will return a graph of the intensity of the selected metric along the *y*-axis versus a time variable along the *x*-axis. The number of top events (n) from the chosen metric may be highlighted in a brighter colour with the aesthetic value colour_n. This function differs in use from lolli_plot in that it must be created as a ggplot2 'geom' object. The benefit of this being that one may add additional information layer by layer to the figure as geoms as necessary.

Usage

```
geom_lolli(mapping = NULL, data = NULL, ..., n = 0, na.rm = FALSE,
  show.legend = NA, inherit.aes = TRUE)
```

Arguments

mapping	Set of aesthetic mappings created by aes() or aes_(). If specified and inherit.aes = TRUE (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.
data	The data to be displayed in this layer. There are three options: 1. If NULL, the default, the data is inherited from the plot data as specified in the call to ggplot(). 2. A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify() for which variables will be created. 3. A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data.
...	other arguments passed on to layer. These are often aesthetics, used to set an aesthetic to a fixed value, like color = "red" or size = 3. They may also be parameters to the paired geom/stat.
n	The number of top events to highlight as based on the value provided to aes(y). Default is 0.
na.rm	If FALSE (the default), removes missing values with a warning. If TRUE silently removes missing values.
show.legend	Logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes. It can also be a named logical vector to finely select the aesthetics to display.
inherit.aes	If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. borders().

Aesthetics

geom_lolli understands the following aesthetics (required aesthetics are in bold):

- x
- y
- alpha
- color
- linetype
- size
- shape
- stroke
- fill
- colour_n While this value may be used as an aesthetic, it works better as a parameter for this function because it is set to use discrete values. One may provide continuous values to colour_n but remember that one may not provide multiple continuous or discrete scales to a single ggplot2 object. Therefore, if one provides a continuous value to aes(colour), the values supplied to colour_n must be discrete. ggplot2 will attempt to do this automatically.

Author(s)

Robert W. Schlegel

See Also

lolli_plot for a non-geom based lolliplot function.

Examples

```
ts <- ts2clm(sst_WA, climatologyPeriod = c("1983-01-01", "2012-12-31"))
res <- detect_event(ts)
mhw <- res$event

library(ggplot2)

# Height of lollis represent event durations and their colours
# are mapped to the events' cumulative intensity:
ggplot(mhw, aes(x = date_peak, y = duration)) +
  geom_lolli(aes(colour = intensity_cumulative)) +
  scale_color_distiller(palette = "Spectral", name = "Cumulative \nintensity") +
  xlab("Date") + ylab("Event duration [days]")

# Height of lollis represent event durations and the top three (longest)
# lollis are highlighted in red:
ggplot(mhw, aes(x = date_peak, y = duration)) +
  geom_lolli(n = 3, colour_n = "red") +
  scale_color_distiller(palette = "Spectral") +
  xlab("Peak date") + ylab("Event duration [days]")
```

```
# Because this is a proper geom, any number of ill-advised things
# may be done with it:
ggplot(mhw, aes(x = event_no, y = intensity_max)) +
  geom_lolli(shape = 5, aes(colour = rate_onset), linetype = "dotted") +
  scale_color_distiller(palette = "RdYlGn", name = "Rate \nonset") +
  xlab("Event number") + ylab("Max intensity [degree C]")
```

`lolli_plot` *Create a timeline of selected event metrics as 'lollipops'.*

Description

Visualise a timeline of several possible event metrics as 'lollipop' graphs.

Usage

```
lolli_plot(data, xaxis = "date_peak", metric = "intensity_max",
  event_count = 3)
```

Arguments

`data`	Output from the `detect_event` function.
`xaxis`	One of `event_no`, `date_start` or `date_peak`. Default is `date_start`.
`metric`	One of `intensity_mean`, `intensity_max`, `intensity_cumulative` and `duration`. Default is `intensity_max`.
`event_count`	The number of top events to highlight, as determined by the value given to `metric`. Default is 3.

Value

The function will return a graph of the intensity of the selected `metric` along the y-axis and the chosen `xaxis` value. The number of top events as per `event_count` will be highlighted in a brighter colour. This function differs in use from `geom_lolli` in that it creates a stand-alone figure. The benefit of this being that one must not have any prior knowledge of `ggplot2` to create the figure.

Author(s)

Albertus J. Smit and Robert W. Schlegel

Examples

```
ts <- ts2clm(sst_WA, climatologyPeriod = c("1983-01-01", "2012-12-31"))
res <- detect_event(ts)

library(ggplot2)

# The default output
lolli_plot(res)
```

`make_whole` *Constructs a continuous, uninterrupted time series of temperatures.*

Description

Takes a series of dates and temperatures, and if irregular (but ordered), inserts missing dates and fills correpsonding temperatures with NAs.

Usage

```
make_whole(data, x = t, y = temp)
```

Arguments

`data`	A data frame with columns for date and temperature data. Ordered daily data are expected, and although missing values (NA) can be accommodated, the function is only recommended when NAs occur infrequently, preferably at no more than 3 consecutive days.
`x`	A column with the daily time vector (see details). For backwards compatibility, the column is named `t` by default.
`y`	A column with the response vector. RmarineHeatWaves version <= 0.15.9 assumed that this would be daily seawater temperatures, but as of version 0.16.0 it may be any arbitrary measurement taken at a daily frequency. The default remains temperature, and the default column name is therefore `temp`, again hopefully ensuring backwards compatibility.

Details

1. Upon import, the package uses ‘zoo‘ and ‘lubridate‘ to process the input date and temperature data. It reads in daily data with the time vector specified as either `POSIXct` or `Date` (e.g. "1982-01-01 02:00:00" or "1982-01-01"). The data may be an irregular time series, but date must be ordered. The function constructs a complete time series from the start date to the end date, and fills in the regions in the time series where temperature data are missing with NAs in the temperature vector. There must only be one temperature value per day otherwise the function takes the mean of multiple values. It is up to the user to calculate daily data from sub-daily measurements. Leap years are automatically accommodated by this function.
2. This function can handle some of missing days, but this is not a licence to actually use these data for the detection of anomalous thermal events. Hobday et al. (2016) recommend gaps of no more than 3 days, which may be adjusted by setting the `maxPadLength` argument of the `ts2clm` function. The longer and more frequent the gaps become the lower the fidelity of the annual climatology and threshold that can be calculated, which will not only have repercussions for the accuracy at which the event metrics can be determined, but also for the number of events that can be detected.
3. It is recommended that a climatology period of at least 30 years is specified in order to capture any decadal thermal periodicities.

4. A faster version of this function called `make_whole_fast` is enabled by default in `ts2clm`, and we assume (hopefully correctly) that the user is certain that her data do not have missing rows (dates) or duplicated values. This slower but more robust function (i.e. `make_whole`) may be selected as an argument to `ts2clm` in case the data may have some issues.

Value

The function will return a data frame with three columns. The column headed `doy` (day-of-year) is the Julian day running from 1 to 366, but modified so that the day-of-year series for non-leap-years runs 1...59 and then 61...366. For leap years the 60th day is February 29. See the example, below. The other two columns take the names of `x` and `y`, if supplied, or it will be `t` and `temp` in case the default values were used. The `x` (or `t`) column is a series of dates of class `Date`, while `y` (or `temp`) is the measured variable. This time series will be uninterrupted and continuous daily values between the first and last dates of the input data.

Author(s)

Smit, A. J.

`make_whole_fast` *Constructs a continuous, uninterrupted time series of temperatures (faster).*

Description

Takes a series of dates and temperatures, and if irregular (but ordered), inserts missing dates and fills correpsonding temperatures with NAs.

Usage

```
make_whole_fast(data)
```

Arguments

`data`	A data frame with columns for date (`ts_x`) and temperature (`ts_y`) data. Ordered daily data are expected, and although missing values (NA) can be accommodated, the function is only recommended when NAs occur infrequently, preferably at no more than three consecutive days.

Details

1. This function reads in daily data with the time vector specified as `Date` (e.g. "1982-01-01").
2. It is up to the user to calculate daily data from sub-daily measurements. Leap years are automatically accommodated by this function.

3. This function can handle some missing days, but this is not a licence to actually use these data for the detection of anomalous thermal events. Hobday et al. (2016) recommend gaps of no more than 3 days, which may be adjusted by setting the `maxPadLength` argument of the `ts2clm` function. The longer and more frequent the gaps become the lower the fidelity of the annual climatology and threshold that can be calculated, which will not only have repercussions for the accuracy at which the event metrics can be determined, but also for the number of events that can be detected.
4. The original `make_whole` tests to see if some rows are duplicated, or if replicate temperature measurements are present per day. In `make_whole_fast` (this function) this has been disabled; also, the latter function lacks the facility to check if the time series is complete and regular (i.e. no missing values in the date vector). Effectively, we now only set up the day-of-year (doy) vector in `make_whole_fast`. Should the user be concerned about the potential for repeated measurements or worry that the time series is irregular, we suggest that the necessary checks and fixes are implemented prior to feeding the time series to `ts2clim` via `make_whole_fast`, or to use `make_whole` instead. For very large gridded temperature records it probably makes a measurable difference if the 'fast' version is used, but it might prevent `detect_event` from failing should some gridded cells contain missing rows or some duplicated values. When using the fast algorithm, we assume that the user has done all the necessary work to ensure that the time vector is regular and without repeated measurements beforehand.

Value

The function will return a data frame with three columns. The column headed `doy` (day-of-year) is the Julian day running from 1 to 366, but modified so that the day-of-year series for non-leap-years runs 1...59 and then 61...366. For leap years the 60th day is February 29. The `ts_x` column is a series of dates of class `Date`, while `y` is the measured variable. This time series will be uninterrupted and continuous daily values between the first and last dates of the input data.

Author(s)

Smit, A. J., Schlegel, R. W.

`na_interp` *Pad NA gaps of user-defined width with interpolated values.*

Description

An internal function that helps to create a time series that will then be used by `clim_calc` within `ts2clm` to produce a climatology as desired by the user.

Usage

```
na_interp(doy = doy, x = ts_x, y = ts_y, maxPadLength)
```

Arguments

doy	Date-of-year as per make_whole or make_whole_fast.
x	Date as per make_whole or make_whole_fast.
y	Measurement variable as per make_whole or make_whole_fast.
maxPadLength	Specifies the maximum length of days over which to interpolate (pad) missing data (specified as NA) in the input temperature time series; i.e., any consecutive blocks of NAs with length greater than maxPadLength will be left as NA. Set as an integer.

Value

The function returns the data (a data.table) in a long format.

Author(s)

Smit, A. J.

---

proto_event *Detect proto-events based on a chosen criterion (column).*

Description

An internal function that detects the events according to the heatwave definition, and joins across the gaps if desired.

Usage

```
proto_event(t_series, criterion_column, minDuration, joinAcrossGaps,
  maxGap)
```

Arguments

t_series	A dataframe of the correct dimensions inherited from detect_event within which this runs.
criterion_column	The column to use for the detection of events.
minDuration	Minimum duration for acceptance of detected events.
joinAcrossGaps	This logic gate tells this internal function if it should connect events across the maxGap (see below). The default it inherits is TRUE.
maxGap	This is the number of rows (days) across which distinct events will be combined into one event if joinAcrossGaps = TRUE.

Value

A dataframe that will be used within detect_event, or which can be returned by detect_event if the switch 'protoEvent' is specified as TRUE.

Author(s)

Albertus J. Smit, Robert W. Schlegel

`smooth_percentile` *Detect the climatology for a time series.*

Description

An internal function that helps to create climatologies in accordance with the Hobday et al. (2016) standard.

Usage

```
smooth_percentile(data, smoothPercentileWidth, var_calc)
```

Arguments

`data`	The data given to this function during the calculations performed by `ts2clm`.
`smoothPercentileWidth`	The width of the smoothing window to be applied. The default is `31` days.
`var_calc`	This is passed from the ts2clm argument `var` and tells the function if a var column exists that needs to be smoothed.

Value

The function returns the data in the same format it was input as, with the climatology values smoothed as desired.

Author(s)

Smit, A. J.

`sst_Med` *Optimally interpolated 0.25 degree SST for the Mediterranean region.*

Description

A dataset containing the sea surface temperature (in degrees Celsius) and date for the Mediterranean region from 1982-01-01 to 2014-12-31.

Usage

```
sst_Med
```

Format

A data frame with 12053 rows and 2 variables:

t date, as POSIXct

temp SST, in degrees Celsius ...

Details

lon/lat: 9/43.5

Source

https://www.ncdc.noaa.gov/oisst

---

sst_NW_Atl *Optimally interpolated 0.25 degree SST for the NW Atlantic region.*

Description

A dataset containing the sea surface temperature (in degrees Celsius) and date for the Northwest Atlantic region from 1982-01-01 to 2014-12-31.

Usage

```
sst_NW_Atl
```

Format

A data frame with 12053 rows and 2 variables:

t date, as POSIXct

temp SST, in degrees Celsius ...

Details

lon/lat: -67/43

Source

https://www.ncdc.noaa.gov/oisst

sst_WA — *Optimally interpolated 0.25 degree SST for the Western Australian region.*

Description

A dataset containing the sea surface temperature temperature (in degrees Celsius) and date for the Western Australian region for the period 1982-01-01 to 2014-12-31.

Usage

```
sst_WA
```

Format

A data frame with 12053 rows and 2 variables:

t date, as POSIXct

temp SST, in degrees Celsius ...

Details

lon/lat: 112.5/-29.5

Source

https://www.ncdc.noaa.gov/oisst

ts2clm — *Make a climatology from a daily time series.*

Description

Creates a daily climatology from a time series of daily temperatures using a user-specified sliding window for the mean and threshold calculation, followed by an optional moving average smoother as used by Hobday et al. (2016).

Usage

```
ts2clm(data, x = t, y = temp, climatologyPeriod, robust = FALSE,
  maxPadLength = FALSE, windowHalfWidth = 5, pctile = 90,
  smoothPercentile = TRUE, smoothPercentileWidth = 31,
  clmOnly = FALSE, var = FALSE)
```

Arguments

data	A data frame with two columns. In the default setting (i.e. ommitting the arguments x and y; see immediately below), the data set is expected to have the headers t and temp. The t column is a vector of dates of class Date, while temp is the measured variable (by default it is assumed to be temperature).
x	This column is expected to contain a vector of dates. If a column headed t is present in the dataframe, this argument may be ommitted; otherwise, specify the name of the column with dates here.
y	This is a column containing the measurement variable. If the column name differs from the default (i.e. temp), specify the name here.
climatologyPeriod	Required. To this argument should be passed two values (see example below). The first value should be the chosen date for the start of the climatology period, and the second value the end date of said period. This chosen period (preferably 30 years in length) is then used to calculate the seasonal cycle and the extreme value threshold.
robust	This argument has been deprecated and no longer has affects how the function operates.
maxPadLength	Specifies the maximum length of days over which to interpolate (pad) missing data (specified as NA) in the input temperature time series; i.e., any consecutive blocks of NAs with length greater than maxPadLength will be left as NA. The default is FALSE. Set as an integer to interpolate. Setting maxPadLength to TRUE will return an error.
windowHalfWidth	Width of sliding window about day-of-year (to one side of the center day-of-year) used for the pooling of values and calculation of climatology and threshold percentile. Default is 5 days, which gives a window width of 11 days centered on the 6th day of the series of 11 days.
pctile	Threshold percentile (%) for detection of events (MHWs). Default is 90th percentile. Should the intent be to use these threshold data for MCSs, set pctile = 10. Or some other low value.
smoothPercentile	Boolean switch selecting whether to smooth the climatology and threshold percentile time series with a moving average of smoothPercentileWidth. Default is TRUE.
smoothPercentileWidth	Full width of moving average window for smoothing climatology and threshold. The default is 31 days.
clmOnly	Choose to calculate and return only the climatologies. The default is FALSE.
var	This argument has been introduced to allow the user to choose if the variance of the seasonal signal per doy should be calculated. The default of FALSE will prevent the calculation, potentially increasing speed of calculations on gridded data and reducing the size of the output. The variance was initially introduced as part of the standard output from Hobday et al. (2016), but few researchers use it and so it is generally regarded now as unnecessary.

Details

1. This function assumes that the input time series consists of continuous daily values with few missing values. Time ranges which start and end part-way through the calendar year are supported.
2. It is recommended that a period of at least 30 years is specified in order to produce a climatology that smooths out any decadal thermal periodicities that may be present. It is further advised that full the start and end dates for the climatology period result in full years, e.g. "1982-01-01" to "2011-12-31" or "1982-07-01" to "2012-06-30"; if not, this may result in an unequal weighting of data belonging with certain months within a time series. A daily climatology will be created; that is, the climatology will be comprised of one mean temperature for each day of the year (365 or 366 days, depending on how leap years are dealt with), and the mean will be based on a sample size that is a function of the length of time determined by the start and end values given to `climatologyPeriod` and the width of the sliding window specified in `windowHalfWidth`.
3. This function supports leap years. This is done by ignoring Feb 29s for the initial calculation of the climatology and threshold. The values for Feb 29 are then linearly interpolated from the values for Feb 28 and Mar 1.
4. Previous versions of `ts2clm()` tested to see if some rows are duplicated, or if replicate temperature readings are present per day, but this has now been disabled. Should the user be concerned about such repeated measurements, we suggest that the necessary checks and fixes are implemented prior to feeding the time series to `ts2clm()`.

The original Python algorithm was written by Eric Oliver, Institute for Marine and Antarctic Studies, University of Tasmania, Feb 2015, and is documented by Hobday et al. (2016).

Value

The function will return a tibble (see the `tidyverse`) with the input time series and the newly calculated climatology. The climatology contains the seasonal climatology and the threshold for calculating MHWs. The software was designed for creating climatologies of daily temperatures, and the units specified below reflect that intended purpose. However, various other kinds of climatologies may be created, and if that is the case, the appropriate units need to be determined by the user.

`doy`	Julian day (day-of-year). For non-leap years it runs 1...59 and 61...366, while leap years run 1...366.
`t`	The date vector in the original time series supplied in `data`. If an alternate column was provided to the `x` argument, that name will rather be used for this column.
`temp`	The measurement vector as per the the original `data` supplied to the function. If a different column was given to the `y` argument that will be shown here.
`seas`	Climatological seasonal cycle [deg. C].
`thresh`	Seasonally varying threshold (e.g., 90th percentile) [deg. C]. This is used in `detect_event` for the detection/calculation of events (MHWs).
`var`	Seasonally varying variance (standard deviation) [deg. C]. This column is not returned if `var = FALSE` (defaut).

Should `clmOnly` be enabled, only the 365 or 366 day climatology will be returned.

Author(s)

Albertus J. Smit, Robert W. Schlegel, Eric C. J. Oliver

References

Hobday, A.J. et al. (2016). A hierarchical approach to defining marine heatwaves, Progress in Oceanography, 141, pp. 227-238, doi:10.1016/j.pocean.2015.12.014

Examples

```
res <- ts2clm(sst_WA, climatologyPeriod = c("1983-01-01", "2012-12-31"))
res[1:10, ]

# Or if one only wants the 366 day climatology
res_clim <- ts2clm(sst_WA, climatologyPeriod = c("1983-01-01", "2012-12-31"),
                   clmOnly = TRUE)
res_clim[1:10, ]

# Or if one wants the variance column included in the results
res_var <- ts2clm(sst_WA, climatologyPeriod = c("1983-01-01", "2012-12-31"),
                  var = TRUE)
res_var[1:10, ]
```

Index