Package 'climdex.pcic'

October 12, 2022

Version 1.1-11

Date 2020-01-21

Title PCIC Implementation of Climdex Routines

Author David Bronaugh <bronaugh@uvic.ca> for the Pacific Climate Impacts Consortium

Maintainer James Hiebert <hiebert@uvic.ca>

Depends R (>= 2.12.0), PCICt (>= 0.5-4)

Imports methods, Rcpp (>= 0.11.4)

Suggests compiler, RUnit

LinkingTo Rcpp

Description PCIC's implementation of Climdex routines for computation of extreme climate indices. Further details on the extreme climate indices can be found at <http://etccdi.pacificclimate.org/list_27_indices.shtml> and in the package manual.

License GPL-3

URL https://www.r-project.org

LazyData yes

BugReports https://github.com/pacificclimate/climdex.pcic/issues/

RoxygenNote 6.0.1

NeedsCompilation yes

Repository CRAN

Date/Publication 2020-01-22 12:30:05 UTC

R topics documented:

climdex.cdd	*The* spells.can.span.years *option specifies whether spells can cross year boundaries – i.e., span years. The default for this is the same as for fclimdex.*

Description

The spells.can.span.years option specifies whether spells can cross year boundaries – i.e., span years. The default for this is the same as for fclimdex.

Usage

```
climdex.cdd(ci, spells.can.span.years = TRUE)
```

Arguments

ci	Object of type climdexInput.
spells.can.span.years	Whether to allow spells to span years

Value

A vector containing the length of the spell for each year.

References

http://etccdi.pacificclimate.org/list_27_indices.shtml

See Also

climdexInput.raw, climdexInput.csv, spell.length.max.

Examples

```
library(PCICt)

## Create a climdexInput object from some data already loaded in and
## ready to go.

## Parse the dates into PCICt.
tmax.dates <- as.PCICt(do.call(paste, ec.1018935.tmax[,c("year",
"jday")]), format="%Y %j", cal="gregorian")
tmin.dates <- as.PCICt(do.call(paste, ec.1018935.tmin[,c("year",
"jday")]), format="%Y %j", cal="gregorian")
prec.dates <- as.PCICt(do.call(paste, ec.1018935.prec[,c("year",
"jday")]), format="%Y %j", cal="gregorian")

## Load the data in.
ci <- climdexInput.raw(ec.1018935.tmax$MAX_TEMP,
```

```
ec.1018935.tmin$MIN_TEMP, ec.1018935.prec$ONE_DAY_PRECIPITATION,
tmax.dates, tmin.dates, prec.dates, base.range=c(1971, 2000))

## Create an annual timeseries of the CDD index.
cdd <- climdex.cdd(ci)
```

`climdex.csdi` *Cold Spell Duration Index*

Description

This function computes the climdex index CSDI.

Usage

```
climdex.csdi(ci, spells.can.span.years = FALSE)
```

Arguments

`ci`	Object of type climdexInput.
`spells.can.span.years`	Whether to allow spells of dry/wet days to span years.

Details

This function takes a climdexInput object as input and computes the climdex index CSDI (Cold Spell Duration Index).

The cold spell duration index is defined as the number of days each year which are part of a "cold spell". A "cold spell" is defined as a sequence of 6 or more days in which the daily minimum temperature is below the 10th percentile of daily minimum temperature for a 5-day running window surrounding this day during the baseline period.

The `spells.can.span.years` option specifies whether spells can cross year boundaries – i.e., span years. The default for this is the same as fclimdex.

Value

A vector containing the value of the index for each year.

Note

These functions may calculate slightly different results than fclimdex.

Behaviour of climdex.wsdi and climdex.csdi differ somewhat from fclimdex. fclimdex considers all days in a warm or cold spell to be part of the year in which the spell ended. climdex.wsdi and climdex.csdi split the spell such that days in each spell are allocated to the separate years in the days occurred.

References

http://etccdi.pacificclimate.org/list_27_indices.shtml

See Also

climdexInput.raw, climdexInput.csv, threshold.exceedance.duration.index.

Examples

```
library(PCICt)

## Create a climdexInput object from some data already loaded in and
## ready to go.

## Parse the dates into PCICt.
tmax.dates <- as.PCICt(do.call(paste, ec.1018935.tmax[,c("year",
"jday")]), format="%Y %j", cal="gregorian")
tmin.dates <- as.PCICt(do.call(paste, ec.1018935.tmin[,c("year",
"jday")]), format="%Y %j", cal="gregorian")
prec.dates <- as.PCICt(do.call(paste, ec.1018935.prec[,c("year",
"jday")]), format="%Y %j", cal="gregorian")

## Load the data in.
ci <- climdexInput.raw(ec.1018935.tmax$MAX_TEMP,
ec.1018935.tmin$MIN_TEMP, ec.1018935.prec$ONE_DAY_PRECIPITATION,
tmax.dates, tmin.dates, prec.dates, base.range=c(1971, 2000))

## Create an annual timeseries of the cold spell duration index.
csdi <- climdex.csdi(ci)
```

`climdex.cwd` *The* `spells.can.span.years` *option specifies whether spells can cross year boundaries – i.e., span years. The default for this is the same as for fclimdex.*

Description

The `spells.can.span.years` option specifies whether spells can cross year boundaries – i.e., span years. The default for this is the same as for fclimdex.

Usage

```
climdex.cwd(ci, spells.can.span.years = TRUE)
```

Arguments

`ci`	Object of type climdexInput.
`spells.can.span.years`	Whether to allow spells to span years

Value

A vector containing the length of the spell for each year.

References

http://etccdi.pacificclimate.org/list_27_indices.shtml

See Also

climdexInput.raw, climdexInput.csv, spell.length.max.

Examples

```
library(PCICt)

## Create a climdexInput object from some data already loaded in and
## ready to go.

## Parse the dates into PCICt.
tmax.dates <- as.PCICt(do.call(paste, ec.1018935.tmax[,c("year",
"jday")]), format="%Y %j", cal="gregorian")
tmin.dates <- as.PCICt(do.call(paste, ec.1018935.tmin[,c("year",
"jday")]), format="%Y %j", cal="gregorian")
prec.dates <- as.PCICt(do.call(paste, ec.1018935.prec[,c("year",
"jday")]), format="%Y %j", cal="gregorian")

## Load the data in.
ci <- climdexInput.raw(ec.1018935.tmax$MAX_TEMP,
ec.1018935.tmin$MIN_TEMP, ec.1018935.prec$ONE_DAY_PRECIPITATION,
tmax.dates, tmin.dates, prec.dates, base.range=c(1971, 2000))

## Create an annual timeseries of the CWD index.
cdd <- climdex.cdd(ci)
```

---

climdex.dtr *Mean Diurnal Temperature Range*

Description

This function computes the diurnal temperature range on a monthly basis.

Usage

```
climdex.dtr(ci, freq = c("monthly", "annual"))
```

Arguments

ci	Object of type climdexInput.
freq	Time frequency to aggregate to.

Details

climdex.dtr computes the mean daily diurnal temperature range. The frequency of observation can be either monthly or annual.

Value

A vector containing the mean monthly or mean annual diurnal temperature range.

Note

This function creates results which may differ in the 3rd decimal place from the results from fclimdex.

References

http://etccdi.pacificclimate.org/list_27_indices.shtml

See Also

climdexInput.raw, climdexInput.csv.

Examples

```
library(PCICt)

## Create a climdexInput object from some data already loaded in and
## ready to go.

## Parse the dates into PCICt.
tmax.dates <- as.PCICt(do.call(paste, ec.1018935.tmax[,c("year",
"jday")]), format="%Y %j", cal="gregorian")
tmin.dates <- as.PCICt(do.call(paste, ec.1018935.tmin[,c("year",
"jday")]), format="%Y %j", cal="gregorian")
prec.dates <- as.PCICt(do.call(paste, ec.1018935.prec[,c("year",
"jday")]), format="%Y %j", cal="gregorian")

## Load the data in.
ci <- climdexInput.raw(ec.1018935.tmax$MAX_TEMP,
ec.1018935.tmin$MIN_TEMP, ec.1018935.prec$ONE_DAY_PRECIPITATION,
tmax.dates, tmin.dates, prec.dates, base.range=c(1971, 2000))

## Create a monthly timeseries of mean diurnal temperature range.
dtr <- climdex.dtr(ci)
```

climdex.fd *Frost Days*

Description

This function computes the climdex index FD.

Usage

```
climdex.fd(ci)
```

Arguments

ci	Object of type climdexInput.

Details

This function takes a climdexInput object as input and computes the FD (frost days) climdex index: that is, the annual count of days where daily minimum temperature drops below 0 degrees Celsius.

Value

A vector containing the number of frost days for each year.

References

http://etccdi.pacificclimate.org/list_27_indices.shtml

See Also

climdexInput.raw, climdexInput.csv.

Examples

```
library(PCICt)

## Create a climdexInput object from some data already loaded in and
## ready to go.

## Parse the dates into PCICt.
tmax.dates <- as.PCICt(do.call(paste, ec.1018935.tmax[,c("year",
"jday")]), format="%Y %j", cal="gregorian")
tmin.dates <- as.PCICt(do.call(paste, ec.1018935.tmin[,c("year",
"jday")]), format="%Y %j", cal="gregorian")
prec.dates <- as.PCICt(do.call(paste, ec.1018935.prec[,c("year",
"jday")]), format="%Y %j", cal="gregorian")

## Load the data in.
ci <- climdexInput.raw(ec.1018935.tmax$MAX_TEMP,
```

```
ec.1018935.tmin$MIN_TEMP, ec.1018935.prec$ONE_DAY_PRECIPITATION,
tmax.dates, tmin.dates, prec.dates, base.range=c(1971, 2000))

## Create an annual timeseries of the number of frost days.
fd <- climdex.fd(ci)
```

`climdex.get.available.indices` *Get available indices by name*

Description

This function returns a vector of (function) names of available indices.

Usage

```
climdex.get.available.indices(ci, function.names = TRUE)
```

Arguments

`ci`	Object of type climdexInput.
`function.names`	Whether to return function names.

Details

This function takes a climdexInput object as input and returns the names of all the indices which may be computed or, if `get.function.names` is TRUE (the default), the names of the functions corresponding to the indices.

Value

A vector containing an annual timeseries of precipitation in wet days.

Examples

```
library(PCICt)

## Create a climdexInput object from some data already loaded in and
## ready to go.

## Parse the dates into PCICt.
tmax.dates <- as.PCICt(do.call(paste, ec.1018935.tmax[,c("year",
"jday")]), format="%Y %j", cal="gregorian")
tmin.dates <- as.PCICt(do.call(paste, ec.1018935.tmin[,c("year",
"jday")]), format="%Y %j", cal="gregorian")
prec.dates <- as.PCICt(do.call(paste, ec.1018935.prec[,c("year",
"jday")]), format="%Y %j", cal="gregorian")

## Load the data in.
```

```
ci <- climdexInput.raw(ec.1018935.tmax$MAX_TEMP,
ec.1018935.tmin$MIN_TEMP, ec.1018935.prec$ONE_DAY_PRECIPITATION,
tmax.dates, tmin.dates, prec.dates, base.range=c(1971, 2000))

## Get list of functions which might be run.
func.names <- climdex.get.available.indices(ci)
```

`climdex.gsl` *Growing Season Length*

Description

This function computes the growing season length (GSL) given the input.

Usage

```
climdex.gsl(ci, gsl.mode = c("GSL", "GSL_first", "GSL_max", "GSL_sum"))
```

Arguments

`ci`	Object of type climdexInput.
`gsl.mode`	Growing season length method to use.

Details

This function takes a climdexInput object as input and computes the growing season length based on this data.

Growing season length as defined by the climdex indices is the number of days between the start of the first spell of warm days in the first half of the year, and the start of the first spell of cold days in the second half of the year. Spells of warm days are defined as six or more days with mean temperature above 5 degrees Celsius; spells of cold days are defined as six or more days with a mean temperature below 5 degrees Celsius.

The three alternate modes provided ('GSL_first', 'GSL_max', and 'GSL_sum') are for testing purposes only. They differ considerably from the first ('GSL') mode. All of them use a list of growing seasons – here defined as six or more consecutive days with a mean temperature greater than or equal to 5 degrees Celsius, followed by either the end of the year or six or more consecutive days with a mean temperature less than 5 degrees Celsius. 'GSL_first' returns the first growing season found; 'GSL_max' returns the longest growing season found; and 'GSL_sum' returns the total length of all growing seasons found.

Value

A vector containing the number of days in the growing season for each year.

Note

Note that fclimdex results may differ from results using the first ('GSL') mode due to bugs in fclimdex. Please ensure you are using the latest version of fclimdex, as there have been numerous bug fixes and the results should, at this point, match.

Please do not use the 'GSL_first', 'GSL_max', or 'GSL_sum' modes for anything other than testing purposes at this time, nor should you rely on this parameter being present in future versions of climdex.pcic.

References

http://etccdi.pacificclimate.org/list_27_indices.shtml

See Also

growing.season.length, climdexInput.csv.

Examples

```
library(PCICt)

## Create a climdexInput object from some data already loaded in and
## ready to go.

## Parse the dates into PCICt.
tmax.dates <- as.PCICt(do.call(paste, ec.1018935.tmax[,c("year",
"jday")]), format="%Y %j", cal="gregorian")
tmin.dates <- as.PCICt(do.call(paste, ec.1018935.tmin[,c("year",
"jday")]), format="%Y %j", cal="gregorian")
prec.dates <- as.PCICt(do.call(paste, ec.1018935.prec[,c("year",
"jday")]), format="%Y %j", cal="gregorian")

## Load the data in.
ci <- climdexInput.raw(ec.1018935.tmax$MAX_TEMP,
ec.1018935.tmin$MIN_TEMP, ec.1018935.prec$ONE_DAY_PRECIPITATION,
tmax.dates, tmin.dates, prec.dates, base.range=c(1971, 2000))

## Create an annual timeseries of the growing season length in days.
gsl <- climdex.gsl(ci)
```

---

climdex.id *Icing Days*

Description

This function computes the climdex index ID.

Usage

```
climdex.id(ci)
```

Arguments

ci	Object of type climdexInput.

Details

This function takes a climdexInput object as input and computes the ID (icing days) climdex index: that is, the annual count of days where daily maximum temperature is below 0 degrees Celsius.

Value

A vector containing the number of icing days for each year.

References

http://etccdi.pacificclimate.org/list_27_indices.shtml

See Also

climdexInput.raw, climdexInput.csv.

Examples

```
library(PCICt)

## Create a climdexInput object from some data already loaded in and
## ready to go.

## Parse the dates into PCICt.
tmax.dates <- as.PCICt(do.call(paste, ec.1018935.tmax[,c("year",
"jday")]), format="%Y %j", cal="gregorian")
tmin.dates <- as.PCICt(do.call(paste, ec.1018935.tmin[,c("year",
"jday")]), format="%Y %j", cal="gregorian")
prec.dates <- as.PCICt(do.call(paste, ec.1018935.prec[,c("year",
"jday")]), format="%Y %j", cal="gregorian")

## Load the data in.
ci <- climdexInput.raw(ec.1018935.tmax$MAX_TEMP,
ec.1018935.tmin$MIN_TEMP, ec.1018935.prec$ONE_DAY_PRECIPITATION,
tmax.dates, tmin.dates, prec.dates, base.range=c(1971, 2000))

## Create an annual timeseries of the number of icing days.
id <- climdex.id(ci)
```

climdex.pcic *climdex.pcic, an implementation of the ETCCDI climate change indices.*

Description

This package implements the ETCCDI's 27 core climate change indices efficiently in R.

Details

The calculation of climate extremes are important in many disciplines. Annual maximum daily precipitation, annual maximum wind speed, and other such extremes are used in many engineering applications. However, they are not as useful when speaking about climate change.

The Expert Team on Climate Change Detection and Indicies (ETCCDI) has created a set of 27 core indices with the intent of capturing the change in the extremes of climate and in selected parameters deemed relevant to other disciplines. These model the following types of parameters:

- Shifts in the number of days where comparatively extreme conditions are observed.
- Growing season length.
- 10th and 90th percentiles of temperature versus baseline conditions.
- Lengths of warm, cold, wet, and dry spells.
- Counts of days where precipitation exceeds a threshold.
- Total precipitation where precipitation exceeds the 95th or 99th percentile of the baseline.

The `climdex.pcic` package provides an implementation of the ETCCDI's 27 core climate change indices. It aims to be reasonably high performance, to handle non-Gregorian calendar types, to be as correct as possible given the definitions of the indices, and to have sufficiently readable and concise code as to facilitate easy verification by inspection.

References

http://etccdi.pacificclimate.org/list_27_indices.shtml

Karl, T.R., N. Nicholls, and A. Ghazi, 1999: CLIVAR/GCOS/WMO workshop on indices and indicators for climate extremes: Workshop summary. Climatic Change, 42, 3-7.

Peterson, T.C., and Coauthors: Report on the Activities of the Working Group on Climate Change Detection and Related Rapporteurs 1998-2001. WMO, Rep. WCDMP-47, WMO-TD 1071, Geneve, Switzerland, 143pp.

Zhang, X., 2005: Avoiding inhomogeneity in percentile-based indices of temperature extremes. Journal of Climate 18.11 (2005):1641-.

See Also

`climdexInput.raw`, `climdexInput.csv`, `climdexInput-class`.

climdex.prcptot *Total Daily Precipitation*

Description

This function computes the climdex index PRCPTOT.

Usage

```
climdex.prcptot(ci)
```

Arguments

ci	Object of type climdexInput.

Details

This function takes a climdexInput object as input and computes the climdex index PRCPTOT: the annual sum of precipitation in wet days (days where precipitation is at least 1mm).

Value

A vector containing an annual timeseries of precipitation in wet days.

References

http://etccdi.pacificclimate.org/list_27_indices.shtml

See Also

climdexInput.raw, climdexInput.csv.

Examples

```
library(PCICt)

## Create a climdexInput object from some data already loaded in and
## ready to go.

## Parse the dates into PCICt.
tmax.dates <- as.PCICt(do.call(paste, ec.1018935.tmax[,c("year",
"jday")]), format="%Y %j", cal="gregorian")
tmin.dates <- as.PCICt(do.call(paste, ec.1018935.tmin[,c("year",
"jday")]), format="%Y %j", cal="gregorian")
prec.dates <- as.PCICt(do.call(paste, ec.1018935.prec[,c("year",
"jday")]), format="%Y %j", cal="gregorian")

## Load the data in.
ci <- climdexInput.raw(ec.1018935.tmax$MAX_TEMP,
```

```
ec.1018935.tmin$MIN_TEMP, ec.1018935.prec$ONE_DAY_PRECIPITATION,
tmax.dates, tmin.dates, prec.dates, base.range=c(1971, 2000))

## Create an annual timeseries of the sum of precipitation in wet days.
prcptot <- climdex.prcptot(ci)
```

`climdex.quantile` *Climdex quantile function*

Description

This function implements R's type=8 in a more efficient manner.

Usage

```
climdex.quantile(x, q = c(0, 0.25, 0.5, 0.75, 1))
```

Arguments

`x`	Data to compute quantiles on.
`q`	Quantiles to be computed.

Details

This is a reimplementation of R's type=8 created to improve the efficiency of this package.

Value

A vector of the quantiles in question.

See Also

`quantile`

Examples

```
## Compute 10th, 50th, and 90th percentile of example data.
climdex.quantile(1:10, c(0.1, 0.5, 0.9))
```

`climdex.r10mm` *Precipitation Exceeding 10mm Per Day*

Description

This function computes the climdex index R10mm.

Usage

```
climdex.r10mm(ci)
```

Arguments

`ci`	Object of type climdexInput.

Details

This function takes a climdexInput object as input and computes the climdex index R10mm: the annual count of days where daily precipitation is more than 10mm per day.

Value

A vector containing the value of the index for each year.

References

http://etccdi.pacificclimate.org/list_27_indices.shtml

See Also

`climdexInput.raw`, `climdexInput.csv`.

Examples

```
library(PCICt)

## Create a climdexInput object from some data already loaded in and
## ready to go.

## Parse the dates into PCICt.
tmax.dates <- as.PCICt(do.call(paste, ec.1018935.tmax[,c("year",
"jday")]), format="%Y %j", cal="gregorian")
tmin.dates <- as.PCICt(do.call(paste, ec.1018935.tmin[,c("year",
"jday")]), format="%Y %j", cal="gregorian")
prec.dates <- as.PCICt(do.call(paste, ec.1018935.prec[,c("year",
"jday")]), format="%Y %j", cal="gregorian")

## Load the data in.
ci <- climdexInput.raw(ec.1018935.tmax$MAX_TEMP,
```

```
ec.1018935.tmin$MIN_TEMP, ec.1018935.prec$ONE_DAY_PRECIPITATION,
tmax.dates, tmin.dates, prec.dates, base.range=c(1971, 2000))

## Create an annual timeseries of the R10mm index.
r10mm <- climdex.r10mm(ci)
```

`climdex.r20mm` *Precipitation Exceeding 20mm Per Day*

Description

This function computes the climdex index R20mm.

Usage

```
climdex.r20mm(ci)
```

Arguments

`ci`	Object of type climdexInput.

Details

This function takes a climdexInput object as input and computes the climdex index R20mm: the annual count of days where daily precipitation is more than 20mm per day.

Value

A vector containing the value of the index for each year.

References

http://etccdi.pacificclimate.org/list_27_indices.shtml

See Also

`climdexInput.raw`, `climdexInput.csv`.

Examples

```
library(PCICt)

## Create a climdexInput object from some data already loaded in and
## ready to go.

## Parse the dates into PCICt.
tmax.dates <- as.PCICt(do.call(paste, ec.1018935.tmax[,c("year",
"jday")]), format="%Y %j", cal="gregorian")
tmin.dates <- as.PCICt(do.call(paste, ec.1018935.tmin[,c("year",
```

```
"jday")]), format="%Y %j", cal="gregorian")
prec.dates <- as.PCICt(do.call(paste, ec.1018935.prec[,c("year",
"jday")]), format="%Y %j", cal="gregorian")

## Load the data in.
ci <- climdexInput.raw(ec.1018935.tmax$MAX_TEMP,
ec.1018935.tmin$MIN_TEMP, ec.1018935.prec$ONE_DAY_PRECIPITATION,
tmax.dates, tmin.dates, prec.dates, base.range=c(1971, 2000))

## Create an annual timeseries of the R20mm index.
r20mm <- climdex.r20mm(ci)
```

`climdex.r95ptot` *Total Daily Precipitation Exceeding 95%ile Threshold*

Description

This function computes the climdex index R95pTOT.

Usage

```
climdex.r95ptot(ci)
```

Arguments

`ci`	Object of type climdexInput.

Details

This function takes a climdexInput object as input and computes the climdex index R95pTOT: the annual sum of precipitation in days where daily precipitation exceeds the 95th percentile of daily precipitation in the base period.

Value

A vector containing an annual timeseries of precipitation exceeding the threshold.

References

http://etccdi.pacificclimate.org/list_27_indices.shtml

See Also

`climdexInput.raw`, `climdexInput.csv`.

Examples

```
library(PCICt)

## Create a climdexInput object from some data already loaded in and
## ready to go.

## Parse the dates into PCICt.
tmax.dates <- as.PCICt(do.call(paste, ec.1018935.tmax[,c("year",
"jday")]), format="%Y %j", cal="gregorian")
tmin.dates <- as.PCICt(do.call(paste, ec.1018935.tmin[,c("year",
"jday")]), format="%Y %j", cal="gregorian")
prec.dates <- as.PCICt(do.call(paste, ec.1018935.prec[,c("year",
"jday")]), format="%Y %j", cal="gregorian")

## Load the data in.
ci <- climdexInput.raw(ec.1018935.tmax$MAX_TEMP,
ec.1018935.tmin$MIN_TEMP, ec.1018935.prec$ONE_DAY_PRECIPITATION,
tmax.dates, tmin.dates, prec.dates, base.range=c(1971, 2000))

## Create an annual timeseries of the R95pTOT index.
r95ptot <- climdex.r95ptot(ci)
```

---

climdex.r99ptot *Total Daily Precipitation Exceeding 99%ile Threshold*

Description

This function computes the climdex index R99pTOT.

Usage

```
climdex.r99ptot(ci)
```

Arguments

ci	Object of type climdexInput.

Details

This function takes a climdexInput object as input and computes the climdex index R99pTOT: the annual sum of precipitation in days where daily precipitation exceeds the 99th percentile of daily precipitation in the base period.

Value

A vector containing an annual timeseries of precipitation exceeding the threshold.

References

http://etccdi.pacificclimate.org/list_27_indices.shtml

See Also

climdexInput.raw, climdexInput.csv.

Examples

```
library(PCICt)

## Create a climdexInput object from some data already loaded in and
## ready to go.

## Parse the dates into PCICt.
tmax.dates <- as.PCICt(do.call(paste, ec.1018935.tmax[,c("year",
"jday")]), format="%Y %j", cal="gregorian")
tmin.dates <- as.PCICt(do.call(paste, ec.1018935.tmin[,c("year",
"jday")]), format="%Y %j", cal="gregorian")
prec.dates <- as.PCICt(do.call(paste, ec.1018935.prec[,c("year",
"jday")]), format="%Y %j", cal="gregorian")

## Load the data in.
ci <- climdexInput.raw(ec.1018935.tmax$MAX_TEMP,
ec.1018935.tmin$MIN_TEMP, ec.1018935.prec$ONE_DAY_PRECIPITATION,
tmax.dates, tmin.dates, prec.dates, base.range=c(1971, 2000))

## Create an annual timeseries of the R99pTOT index.
r99ptot <- climdex.r99ptot(ci)
```

---

climdex.rnnmm *Precipitation Exceeding A Specified Amount Per Day*

Description

This function computes the climdex index Rnnmm.

Usage

```
climdex.rnnmm(ci, threshold = 1)
```

Arguments

ci	Object of type climdexInput.
threshold	The threshold to be used for Rnnmm.

Details

This function takes a climdexInput object as input and computes the climdex index Rnnmm: the annual count of days where daily precipitation is more than nn mm per day.

Value

A vector containing the value of the index for each year.

References

http://etccdi.pacificclimate.org/list_27_indices.shtml

See Also

climdexInput.raw, climdexInput.csv.

Examples

```
library(PCICt)

## Create a climdexInput object from some data already loaded in and
## ready to go.

## Parse the dates into PCICt.
tmax.dates <- as.PCICt(do.call(paste, ec.1018935.tmax[,c("year",
"jday")]), format="%Y %j", cal="gregorian")
tmin.dates <- as.PCICt(do.call(paste, ec.1018935.tmin[,c("year",
"jday")]), format="%Y %j", cal="gregorian")
prec.dates <- as.PCICt(do.call(paste, ec.1018935.prec[,c("year",
"jday")]), format="%Y %j", cal="gregorian")

## Load the data in.
ci <- climdexInput.raw(ec.1018935.tmax$MAX_TEMP,
ec.1018935.tmin$MIN_TEMP, ec.1018935.prec$ONE_DAY_PRECIPITATION,
tmax.dates, tmin.dates, prec.dates, base.range=c(1971, 2000))

## Create an annual timeseries of the R1mm index.
rnnmm <- climdex.rnnmm(ci)
```

---

climdex.rx1day *Monthly Maximum 1-day Precipitation*

Description

This function computes the climdex index Rx1day.

Usage

```
climdex.rx1day(ci, freq = c("monthly", "annual"))
```

Arguments

ci	Object of type climdexInput.
freq	Time frequency to aggregate to.

Details

This function takes a climdexInput object as input and computes the climdex index Rx1day: monthly or annual maximum 1-day precipitation.

Value

A vector containing the value of the index for each month of each year.

Note

The default behaviour of climdex.rx5day differs somewhat from fclimdex, as fclimdex and climdex.pcic differ on the definition of Rx5day. The running sum series computed by fclimdex is off by 2 days, and the first day a running sum can be computed for is left out entirely. The behaviour of fclimdex can be replicated by setting center.mean.on.last.day to TRUE.

References

http://etccdi.pacificclimate.org/list_27_indices.shtml

See Also

climdexInput.raw, climdexInput.csv.

Examples

```
library(PCICt)

## Create a climdexInput object from some data already loaded in and
## ready to go.

## Parse the dates into PCICt.
tmax.dates <- as.PCICt(do.call(paste, ec.1018935.tmax[,c("year",
"jday")]), format="%Y %j", cal="gregorian")
tmin.dates <- as.PCICt(do.call(paste, ec.1018935.tmin[,c("year",
"jday")]), format="%Y %j", cal="gregorian")
prec.dates <- as.PCICt(do.call(paste, ec.1018935.prec[,c("year",
"jday")]), format="%Y %j", cal="gregorian")

## Load the data in.
ci <- climdexInput.raw(ec.1018935.tmax$MAX_TEMP,
ec.1018935.tmin$MIN_TEMP, ec.1018935.prec$ONE_DAY_PRECIPITATION,
tmax.dates, tmin.dates, prec.dates, base.range=c(1971, 2000))

## Create a timeseries of monthly maximum 1-day precipitation.
rx1day <- climdex.rx1day(ci)
```

`climdex.rx5day` *Monthly Maximum 5-day Consecutive Precipitation*

Description

This function computes the climdex index Rx5day.

Usage

```
climdex.rx5day(ci, freq = c("monthly", "annual"),
  center.mean.on.last.day = FALSE)
```

Arguments

`ci`	Object of type climdexInput.
`freq`	Time frequency to aggregate to.
`center.mean.on.last.day`	Whether to center the 5-day running mean on the last day of the window, instead of the center day.

Details

This function takes a climdexInput object as input and computes the climdex index Rx5day: monthly or annual maximum 5-day consecutive precipitation.

Value

A vector containing the value of the index for each month of each year.

Note

The default behaviour of climdex.rx5day differs somewhat from fclimdex, as fclimdex and climdex.pcic differ on the definition of Rx5day. The running sum series computed by fclimdex is off by 2 days, and the first day a running sum can be computed for is left out entirely. The behaviour of fclimdex can be replicated by setting center.mean.on.last.day to TRUE.

References

http://etccdi.pacificclimate.org/list_27_indices.shtml

See Also

`climdexInput.raw`, `climdexInput.csv`.

Examples

```
library(PCICt)

## Create a climdexInput object from some data already loaded in and
## ready to go.

## Parse the dates into PCICt.
tmax.dates <- as.PCICt(do.call(paste, ec.1018935.tmax[,c("year",
"jday")]), format="%Y %j", cal="gregorian")
tmin.dates <- as.PCICt(do.call(paste, ec.1018935.tmin[,c("year",
"jday")]), format="%Y %j", cal="gregorian")
prec.dates <- as.PCICt(do.call(paste, ec.1018935.prec[,c("year",
"jday")]), format="%Y %j", cal="gregorian")

## Load the data in.
ci <- climdexInput.raw(ec.1018935.tmax$MAX_TEMP,
ec.1018935.tmin$MIN_TEMP, ec.1018935.prec$ONE_DAY_PRECIPITATION,
tmax.dates, tmin.dates, prec.dates, base.range=c(1971, 2000))

## Create a timeseries of monthly maximum 5-day consecutive precipitation.
rx5day <- climdex.rx5day(ci)
```

`climdex.sdii` *Simple Precpitation Intensity Index*

Description

This function computes the climdex index SDII.

Usage

```
climdex.sdii(ci)
```

Arguments

`ci`	Object of type climdexInput.

Details

`climdex.sdii` computes the climdex index SDII, or Simple Precipitation Intensity Index. This is defined as the sum of precipitation in wet days (days with preciptitation over 1mm) during the year divided by the number of wet days in the year.

Value

A vector containing the value of the index for each year.

Note

fclimdex rounds to 1 decimal place, whereas climdex.sdii does not. This results in some small differences.

References

http://etccdi.pacificclimate.org/list_27_indices.shtml

See Also

climdexInput.raw, climdexInput.csv.

Examples

```
library(PCICt)

## Create a climdexInput object from some data already loaded in and
## ready to go.

## Parse the dates into PCICt.
tmax.dates <- as.PCICt(do.call(paste, ec.1018935.tmax[,c("year",
"jday")]), format="%Y %j", cal="gregorian")
tmin.dates <- as.PCICt(do.call(paste, ec.1018935.tmin[,c("year",
"jday")]), format="%Y %j", cal="gregorian")
prec.dates <- as.PCICt(do.call(paste, ec.1018935.prec[,c("year",
"jday")]), format="%Y %j", cal="gregorian")

## Load the data in.
ci <- climdexInput.raw(ec.1018935.tmax$MAX_TEMP,
ec.1018935.tmin$MIN_TEMP, ec.1018935.prec$ONE_DAY_PRECIPITATION,
tmax.dates, tmin.dates, prec.dates, base.range=c(1971, 2000))

## Create a timeseries of annual SDII values.
sdii <- climdex.sdii(ci)
```

---

climdex.su *Summer Days*

Description

This function computes the climdex index SU.

Usage

```
climdex.su(ci)
```

Arguments

ci	Object of type climdexInput.

Details

This function takes a climdexInput object as input and computes the SU (summer days) climdex index: that is, the annual count of days where daily maximum temperature exceeds 25 degrees Celsius.

Value

A vector containing the number of summer days for each year.

References

http://etccdi.pacificclimate.org/list_27_indices.shtml

See Also

climdexInput.raw, climdexInput.csv.

Examples

```
library(PCICt)

## Create a climdexInput object from some data already loaded in and
## ready to go.

## Parse the dates into PCICt.
tmax.dates <- as.PCICt(do.call(paste, ec.1018935.tmax[,c("year",
"jday")]), format="%Y %j", cal="gregorian")
tmin.dates <- as.PCICt(do.call(paste, ec.1018935.tmin[,c("year",
"jday")]), format="%Y %j", cal="gregorian")
prec.dates <- as.PCICt(do.call(paste, ec.1018935.prec[,c("year",
"jday")]), format="%Y %j", cal="gregorian")

## Load the data in.
ci <- climdexInput.raw(ec.1018935.tmax$MAX_TEMP,
ec.1018935.tmin$MIN_TEMP, ec.1018935.prec$ONE_DAY_PRECIPITATION,
tmax.dates, tmin.dates, prec.dates, base.range=c(1971, 2000))

## Create an annual timeseries of the number of summer days.
su <- climdex.su(ci)
```

---

climdex.tn10p *Computation of these percentiles involves use of a boostrap procedure, described below but described in more depth in [Zhang, 2005].*

Description

Computation of these values outside of the base period involves comparing the temperature data for each day with the corresponding percentiles for a 5 day running window surrounding that day. The resulting monthly series is then the monthly percentage of values that meet the criteria.

Usage

```
climdex.tn10p(ci, freq = c("monthly", "annual"))
```

Arguments

ci	Object of type climdexInput.
freq	Time frequency to aggregate to.

Details

Computation of these values inside the base period is more complicated. It involves comparison of the daily temperature data with the corresponding day of temperature data in each of (n - 1) sets of data. The sets consist of the data for the base period with the current year replaced with each of the other years. The results of these comparisons are then averaged to give a value between 0 and 1. Finally, the resulting daily series is aggregated to a monthly series by averaging these daily values and multiplying by 100 to give a monthly percentile value.

Value

A vector containing a timeseries containing values of the index on a monthly or annual timescale.

Note

These functions may calculate slightly different results than fclimdex.

The bootstrapping method is not well defined for cases where the base data contains numerous missing values. Because of that, this code (and fclimdex) are not very robust against missing values with respect to these indicies. When computing percentiles inside the base period, both this implementation and fclimdex do not divide through by the number of non-missing values when aggregating the values inside the base period. Instead, they divide through by the number of base years minus one. This will result in a negative bias when missing values are present.

References

http://etccdi.pacificclimate.org/list_27_indices.shtml

See Also

climdexInput.raw, climdexInput.csv.

Examples

```
library(PCICt)

## Create a climdexInput object from some data already loaded in and
## ready to go.

## Parse the dates into PCICt.
tmax.dates <- as.PCICt(do.call(paste, ec.1018935.tmax[,c("year",
"jday")]), format="%Y %j", cal="gregorian")
tmin.dates <- as.PCICt(do.call(paste, ec.1018935.tmin[,c("year",
```

```
"jday")]), format="%Y %j", cal="gregorian")
prec.dates <- as.PCICt(do.call(paste, ec.1018935.prec[,c("year",
"jday")]), format="%Y %j", cal="gregorian")

## Load the data in.
ci <- climdexInput.raw(ec.1018935.tmax$MAX_TEMP,
ec.1018935.tmin$MIN_TEMP, ec.1018935.prec$ONE_DAY_PRECIPITATION,
tmax.dates, tmin.dates, prec.dates, base.range=c(1971, 2000))

## Create a monthly timeseries of the TN10p index.
tn10p <- climdex.tn10p(ci)
```

`climdex.tn90p` *Computation of these percentiles involves use of a boostrap procedure, described below but described in more depth in [Zhang, 2005].*

Description

Computation of these values outside of the base period involves comparing the temperature data for each day with the corresponding percentiles for a 5 day running window surrounding that day. The resulting monthly series is then the monthly percentage of values that meet the criteria.

Usage

```
climdex.tn90p(ci, freq = c("monthly", "annual"))
```

Arguments

`ci`	Object of type climdexInput.
`freq`	Time frequency to aggregate to.

Details

Computation of these values inside the base period is more complicated. It involves comparison of the daily temperature data with the corresponding day of temperature data in each of (n - 1) sets of data. The sets consist of the data for the base period with the current year replaced with each of the other years. The results of these comparisons are then averaged to give a value between 0 and 1. Finally, the resulting daily series is aggregated to a monthly series by averaging these daily values and multiplying by 100 to give a monthly percentile value.

Value

A vector containing a timeseries containing values of the index on a monthly or annual timescale.

Note

These functions may calculate slightly different results than fclimdex.

The bootstrapping method is not well defined for cases where the base data contains numerous missing values. Because of that, this code (and fclimdex) are not very robust against missing values with respect to these indicies. When computing percentiles inside the base period, both this implementation and fclimdex do not divide through by the number of non-missing values when aggregating the values inside the base period. Instead, they divide through by the number of base years minus one. This will result in a negative bias when missing values are present.

References

http://etccdi.pacificclimate.org/list_27_indices.shtml

See Also

climdexInput.raw, climdexInput.csv.

Examples

```
library(PCICt)

## Create a climdexInput object from some data already loaded in and
## ready to go.

## Parse the dates into PCICt.
tmax.dates <- as.PCICt(do.call(paste, ec.1018935.tmax[,c("year",
"jday")]), format="%Y %j", cal="gregorian")
tmin.dates <- as.PCICt(do.call(paste, ec.1018935.tmin[,c("year",
"jday")]), format="%Y %j", cal="gregorian")
prec.dates <- as.PCICt(do.call(paste, ec.1018935.prec[,c("year",
"jday")]), format="%Y %j", cal="gregorian")

## Load the data in.
ci <- climdexInput.raw(ec.1018935.tmax$MAX_TEMP,
ec.1018935.tmin$MIN_TEMP, ec.1018935.prec$ONE_DAY_PRECIPITATION,
tmax.dates, tmin.dates, prec.dates, base.range=c(1971, 2000))

## Create a monthly timeseries of the TN90p index.
tn90p <- climdex.tn90p(ci)
```

---

climdex.tnn *Monthly Minimum of Daily Minimum Temperature*

Description

This function computes the climdex index TNn.

Usage

```
climdex.tnn(ci, freq = c("monthly", "annual"))
```

Arguments

ci	Object of type climdexInput.
freq	Time frequency to aggregate to.

Details

This function takes a climdexInput object as input and computes the monthly or annual minimum of daily minimum temperature.

Value

A vector containing the value of the index for each month.

References

http://etccdi.pacificclimate.org/list_27_indices.shtml

See Also

climdexInput.raw, climdexInput.csv.

Examples

```
library(PCICt)

## Create a climdexInput object from some data already loaded in and
## ready to go.

## Parse the dates into PCICt.
tmax.dates <- as.PCICt(do.call(paste, ec.1018935.tmax[,c("year",
"jday")]), format="%Y %j", cal="gregorian")
tmin.dates <- as.PCICt(do.call(paste, ec.1018935.tmin[,c("year",
"jday")]), format="%Y %j", cal="gregorian")
prec.dates <- as.PCICt(do.call(paste, ec.1018935.prec[,c("year",
"jday")]), format="%Y %j", cal="gregorian")

## Load the data in.
ci <- climdexInput.raw(ec.1018935.tmax$MAX_TEMP,
ec.1018935.tmin$MIN_TEMP, ec.1018935.prec$ONE_DAY_PRECIPITATION,
tmax.dates, tmin.dates, prec.dates, base.range=c(1971, 2000))

## Create a monthly timeseries of minimum daily minimum temperature.
tnn <- climdex.tnn(ci)
```

climdex.tnx *Monthly Maximum of Daily Minimum Temperature*

Description

This function computes the climdex index TNx.

Usage

```
climdex.tnx(ci, freq = c("monthly", "annual"))
```

Arguments

ci	Object of type climdexInput.
freq	Time frequency to aggregate to.

Details

This function takes a climdexInput object as input and computes the monthly or annual maximum of daily minimum temperature.

Value

A vector containing the value of the index for each month.

References

http://etccdi.pacificclimate.org/list_27_indices.shtml

See Also

climdexInput.raw, climdexInput.csv.

Examples

```
library(PCICt)

## Create a climdexInput object from some data already loaded in and
## ready to go.

## Parse the dates into PCICt.
tmax.dates <- as.PCICt(do.call(paste, ec.1018935.tmax[,c("year",
"jday")]), format="%Y %j", cal="gregorian")
tmin.dates <- as.PCICt(do.call(paste, ec.1018935.tmin[,c("year",
"jday")]), format="%Y %j", cal="gregorian")
prec.dates <- as.PCICt(do.call(paste, ec.1018935.prec[,c("year",
"jday")]), format="%Y %j", cal="gregorian")

## Load the data in.
```

```
ci <- climdexInput.raw(ec.1018935.tmax$MAX_TEMP,
ec.1018935.tmin$MIN_TEMP, ec.1018935.prec$ONE_DAY_PRECIPITATION,
tmax.dates, tmin.dates, prec.dates, base.range=c(1971, 2000))

## Create a monthly timeseries of maximum daily minimum temperature.
tnx <- climdex.tnx(ci)
```

`climdex.tr` *Tropical Nights*

Description

This function computes the climdex index TR.

Usage

```
climdex.tr(ci)
```

Arguments

`ci`	Object of type climdexInput.

Details

This function takes a climdexInput object as input and computes the TR (tropical nights) climdex index: that is, the annual count of days where daily minimum temperature stays above 20 degrees Celsius.

Value

A vector containing the number of frost days for each year.

References

http://etccdi.pacificclimate.org/list_27_indices.shtml

See Also

`climdexInput.raw`, `climdexInput.csv`.

Examples

```
library(PCICt)

## Create a climdexInput object from some data already loaded in and
## ready to go.

## Parse the dates into PCICt.
tmax.dates <- as.PCICt(do.call(paste, ec.1018935.tmax[,c("year",
```

```
"jday")]), format="%Y %j", cal="gregorian")
tmin.dates <- as.PCICt(do.call(paste, ec.1018935.tmin[,c("year",
"jday")]), format="%Y %j", cal="gregorian")
prec.dates <- as.PCICt(do.call(paste, ec.1018935.prec[,c("year",
"jday")]), format="%Y %j", cal="gregorian")

## Load the data in.
ci <- climdexInput.raw(ec.1018935.tmax$MAX_TEMP,
ec.1018935.tmin$MIN_TEMP, ec.1018935.prec$ONE_DAY_PRECIPITATION,
tmax.dates, tmin.dates, prec.dates, base.range=c(1971, 2000))

## Create an annual timeseries of the number of tropical nights.
tr <- climdex.tr(ci)
```

`climdex.tx10p` *Computation of these percentiles involves use of a boostrap procedure, described below but described in more depth in [Zhang, 2005].*

Description

Computation of these values outside of the base period involves comparing the temperature data for each day with the corresponding percentiles for a 5 day running window surrounding that day. The resulting monthly series is then the monthly percentage of values that meet the criteria.

Usage

```
climdex.tx10p(ci, freq = c("monthly", "annual"))
```

Arguments

`ci`	Object of type climdexInput.
`freq`	Time frequency to aggregate to.

Details

Computation of these values inside the base period is more complicated. It involves comparison of the daily temperature data with the corresponding day of temperature data in each of (n - 1) sets of data. The sets consist of the data for the base period with the current year replaced with each of the other years. The results of these comparisons are then averaged to give a value between 0 and 1. Finally, the resulting daily series is aggregated to a monthly series by averaging these daily values and multiplying by 100 to give a monthly percentile value.

Value

A vector containing a timeseries containing values of the index on a monthly or annual timescale.

Note

These functions may calculate slightly different results than fclimdex.

The bootstrapping method is not well defined for cases where the base data contains numerous missing values. Because of that, this code (and fclimdex) are not very robust against missing values with respect to these indicies. When computing percentiles inside the base period, both this implementation and fclimdex do not divide through by the number of non-missing values when aggregating the values inside the base period. Instead, they divide through by the number of base years minus one. This will result in a negative bias when missing values are present.

References

http://etccdi.pacificclimate.org/list_27_indices.shtml

See Also

climdexInput.raw, climdexInput.csv.

Examples

```
library(PCICt)

## Create a climdexInput object from some data already loaded in and
## ready to go.

## Parse the dates into PCICt.
tmax.dates <- as.PCICt(do.call(paste, ec.1018935.tmax[,c("year",
"jday")]), format="%Y %j", cal="gregorian")
tmin.dates <- as.PCICt(do.call(paste, ec.1018935.tmin[,c("year",
"jday")]), format="%Y %j", cal="gregorian")
prec.dates <- as.PCICt(do.call(paste, ec.1018935.prec[,c("year",
"jday")]), format="%Y %j", cal="gregorian")

## Load the data in.
ci <- climdexInput.raw(ec.1018935.tmax$MAX_TEMP,
ec.1018935.tmin$MIN_TEMP, ec.1018935.prec$ONE_DAY_PRECIPITATION,
tmax.dates, tmin.dates, prec.dates, base.range=c(1971, 2000))

## Create a monthly timeseries of the TX10p index.
tx10p <- climdex.tx10p(ci)
```

`climdex.tx90p` *Computation of these percentiles involves use of a boostrap procedure, described below but described in more depth in [Zhang, 2005].*

Description

Computation of these values outside of the base period involves comparing the temperature data for each day with the corresponding percentiles for a 5 day running window surrounding that day. The resulting monthly series is then the monthly percentage of values that meet the criteria.

Usage

```
climdex.tx90p(ci, freq = c("monthly", "annual"))
```

Arguments

ci	Object of type climdexInput.
freq	Time frequency to aggregate to.

Details

Computation of these values inside the base period is more complicated. It involves comparison of the daily temperature data with the corresponding day of temperature data in each of (n - 1) sets of data. The sets consist of the data for the base period with the current year replaced with each of the other years. The results of these comparisons are then averaged to give a value between 0 and 1. Finally, the resulting daily series is aggregated to a monthly series by averaging these daily values and multiplying by 100 to give a monthly percentile value.

Value

A vector containing a timeseries containing values of the index on a monthly or annual timescale.

Note

These functions may calculate slightly different results than fclimdex.

The bootstrapping method is not well defined for cases where the base data contains numerous missing values. Because of that, this code (and fclimdex) are not very robust against missing values with respect to these indicies. When computing percentiles inside the base period, both this implementation and fclimdex do not divide through by the number of non-missing values when aggregating the values inside the base period. Instead, they divide through by the number of base years minus one. This will result in a negative bias when missing values are present.

References

http://etccdi.pacificclimate.org/list_27_indices.shtml

See Also

climdexInput.raw, climdexInput.csv.

Examples

```
library(PCICt)

## Create a climdexInput object from some data already loaded in and
## ready to go.

## Parse the dates into PCICt.
tmax.dates <- as.PCICt(do.call(paste, ec.1018935.tmax[,c("year",
"jday")]), format="%Y %j", cal="gregorian")
tmin.dates <- as.PCICt(do.call(paste, ec.1018935.tmin[,c("year",
```

```
"jday")]), format="%Y %j", cal="gregorian")
prec.dates <- as.PCICt(do.call(paste, ec.1018935.prec[,c("year",
"jday")]), format="%Y %j", cal="gregorian")

## Load the data in.
ci <- climdexInput.raw(ec.1018935.tmax$MAX_TEMP,
ec.1018935.tmin$MIN_TEMP, ec.1018935.prec$ONE_DAY_PRECIPITATION,
tmax.dates, tmin.dates, prec.dates, base.range=c(1971, 2000))

## Create a monthly timeseries of the TX90p index.
tx90p <- climdex.tx90p(ci)
```

`climdex.txn` *Monthly Minimum of Daily Maximum Temperature*

Description

This function computes the climdex index TXn.

Usage

```
climdex.txn(ci, freq = c("monthly", "annual"))
```

Arguments

`ci`	Object of type climdexInput.
`freq`	Time frequency to aggregate to.

Details

This function takes a climdexInput object as input and computes the monthly or annual minimum of daily maximum temperature.

Value

A vector containing the value of the index for each month.

References

http://etccdi.pacificclimate.org/list_27_indices.shtml

See Also

`climdexInput.raw`, `climdexInput.csv`.

Examples

```
library(PCICt)

## Create a climdexInput object from some data already loaded in and
## ready to go.

## Parse the dates into PCICt.
tmax.dates <- as.PCICt(do.call(paste, ec.1018935.tmax[,c("year",
"jday")]), format="%Y %j", cal="gregorian")
tmin.dates <- as.PCICt(do.call(paste, ec.1018935.tmin[,c("year",
"jday")]), format="%Y %j", cal="gregorian")
prec.dates <- as.PCICt(do.call(paste, ec.1018935.prec[,c("year",
"jday")]), format="%Y %j", cal="gregorian")

## Load the data in.
ci <- climdexInput.raw(ec.1018935.tmax$MAX_TEMP,
ec.1018935.tmin$MIN_TEMP, ec.1018935.prec$ONE_DAY_PRECIPITATION,
tmax.dates, tmin.dates, prec.dates, base.range=c(1971, 2000))

## Create a monthly timeseries of minimum daily maximum temperature.
txn <- climdex.txn(ci)
```

`climdex.txx` *Monthly Maximum of Daily Maximum Temperature*

Description

This function computes the climdex index TXx.

Usage

```
climdex.txx(ci, freq = c("monthly", "annual"))
```

Arguments

`ci`	Object of type climdexInput.
`freq`	Time frequency to aggregate to.

Details

This function takes a climdexInput object as input and computes the monthly or annual maximum of daily maximum temperature.

Value

A vector containing the value of the index for each month.

References

http://etccdi.pacificclimate.org/list_27_indices.shtml

See Also

climdexInput.raw, climdexInput.csv.

Examples

```
library(PCICt)

## Create a climdexInput object from some data already loaded in and
## ready to go.

## Parse the dates into PCICt.
tmax.dates <- as.PCICt(do.call(paste, ec.1018935.tmax[,c("year",
"jday")]), format="%Y %j", cal="gregorian")
tmin.dates <- as.PCICt(do.call(paste, ec.1018935.tmin[,c("year",
"jday")]), format="%Y %j", cal="gregorian")
prec.dates <- as.PCICt(do.call(paste, ec.1018935.prec[,c("year",
"jday")]), format="%Y %j", cal="gregorian")

## Load the data in.
ci <- climdexInput.raw(ec.1018935.tmax$MAX_TEMP,
ec.1018935.tmin$MIN_TEMP, ec.1018935.prec$ONE_DAY_PRECIPITATION,
tmax.dates, tmin.dates, prec.dates, base.range=c(1971, 2000))

## Create a monthly timeseries of maximum daily maximum temperature.
txx <- climdex.txx(ci)
```

---

climdex.wsdi *Warm Spell Duration Index*

Description

This function computes the climdex index WSDI.

Usage

```
climdex.wsdi(ci, spells.can.span.years = FALSE)
```

Arguments

ci	Object of type climdexInput.
spells.can.span.years	Whether to allow spells of dry/wet days to span years.

Details

This function takes a climdexInput object as input and computes the climdex index WSDI (Warm Spell Duration Index).

The warm spell duration index is defined as the number of days each year which are part of a "warm spell". A "warm spell" is defined as a sequence of 6 or more days in which the daily maximum temperature exceeds the 90th percentile of daily maximum temperature for a 5-day running window surrounding this day during the baseline period.

The `spells.can.span.years` option specifies whether spells can cross year boundaries – i.e., span years. The default for this is the same as fclimdex.

Value

A vector containing the value of the index for each year.

Note

These functions may calculate slightly different results than fclimdex.

Behaviour of climdex.wsdi and climdex.csdi differ somewhat from fclimdex. fclimdex considers all days in a warm or cold spell to be part of the year in which the spell ended. climdex.wsdi and climdex.csdi split the spell such that days in each spell are allocated to the separate years in the days occurred.

References

http://etccdi.pacificclimate.org/list_27_indices.shtml

See Also

`climdexInput.raw`, `climdexInput.csv`, `threshold.exceedance.duration.index`.

Examples

```
library(PCICt)

## Create a climdexInput object from some data already loaded in and
## ready to go.

## Parse the dates into PCICt.
tmax.dates <- as.PCICt(do.call(paste, ec.1018935.tmax[,c("year",
"jday")]), format="%Y %j", cal="gregorian")
tmin.dates <- as.PCICt(do.call(paste, ec.1018935.tmin[,c("year",
"jday")]), format="%Y %j", cal="gregorian")
prec.dates <- as.PCICt(do.call(paste, ec.1018935.prec[,c("year",
"jday")]), format="%Y %j", cal="gregorian")

## Load the data in.
ci <- climdexInput.raw(ec.1018935.tmax$MAX_TEMP,
ec.1018935.tmin$MIN_TEMP, ec.1018935.prec$ONE_DAY_PRECIPITATION,
tmax.dates, tmin.dates, prec.dates, base.range=c(1971, 2000))
```

```
## Create an annual timeseries of the warm spell duration index.
wsdi <- climdex.wsdi(ci)
```

`climdexInput` *climdexInput*

Description

The climdexInput class contains all the data necessary to compute the climdex indices.

Details

The `climdexInput` class consists of all the data necessary to compute the climdex indices. Users will not need to modify any of the slots in this class. That being said, users may want or need to repurpose this data for further analysis. The following description of the data is aimed at that audience.

The `data` slot contains time series' of daily data of equal length for each of the provided variables. Missing days have been replaced with NA. The `dates` slot is the corresponding series of dates (of type PCICt) for the daily data.

The `quantiles` slot contains quantiles used for computing the tn/tx 10/90p indices, w/csdi, r95ptot, and r99ptot. If precipitation data is supplied, the 'prec' member contains the 95th and 99th percentile values for precipitation within the base period. For tmin and tmax, if present each will have a corresponding member in the slot. Within each of these, there will be an 'inbase' and 'outbase' member, corresponding to thresholds to be used within the base period (inbase) and outside the base period (outbase). The 'inbase' member consists of one percentile for each day of the year, computed using an n-day (default is 5-day) running window surrounding that day. These percentiles are computed for at least the 10th and 90th percentile of the data. For the 'outbase' member, given n years of data to use as the base period, there are n * (n - 1) sets of daily quantiles of the same type as those in 'inbase'.

To ease computation of monthly and annual data, `date.factors` contains date factors which group data into annual and monthly time buckets. They are of the same length as the time series and can be reused for computation of any annual or monthly aggregates.

The climdexInput class also includes NA masks for both monthly and annual as parts of the `namasks` slot. Each of these masks consist of a vector of numbers of the same length as the monthly or annual output data. The values used are 1 to signify that the data meets the QC criteria, and NA to signify it does not. Years with more than (by default) 15 days missing, and months with more than (by default) 3 days missing, are considered to be of poor quality and are masked here with NA. These thresholds can be set when instantiating the object, and are stored in the `max.missing.days` slot.

The `base.range` slot contains vector of type PCICt containing the first and last day included in the baseline.

The `northern.hemisphere` slot contains a boolean indicating whether the data came from the northern hemisphere. If FALSE, data is assumed to have come from the southern hemisphere. This is used when computing growing season length; if the data is from the southern hemisphere, growing season length is the growing season starting in the beginning of July of the year indicated, running to the end of June of the following year.

The max.missing.days slot is a vector consisting of 'annual' (the number of days that can be missing in a year) and 'monthly' (the number of days that can be missing in a month. If one month in a year fails the test, the corresponding year will be omitted.

Slots

data Time series of supplied data variables.

quantiles Threshold quantiles used for threshold-based indices.

namasks Data quality masks for annual and monthly data.

dates Date sequence (type PCICt) corresponding to temperature and precipitation data.

jdays Julian days for the date sequence.

base.range Date range (type PCICt) of baseline period.

date.factors Factors used for creation of annual and monthly indices.

northern.hemisphere Boolean used when computing growing season length.

max.missing.days Maximum number of missing days of data for annual and monthly data.

See Also

climdexInput.csv, climdexInput.raw.

Examples

```
library(PCICt)

## Parse the dates into PCICt.
tmax.dates <- as.PCICt(do.call(paste, ec.1018935.tmax[,c("year",
"jday")]), format="%Y %j", cal="gregorian")
tmin.dates <- as.PCICt(do.call(paste, ec.1018935.tmin[,c("year",
"jday")]), format="%Y %j", cal="gregorian")
prec.dates <- as.PCICt(do.call(paste, ec.1018935.prec[,c("year",
"jday")]), format="%Y %j", cal="gregorian")

## Load the data in.
ci <- climdexInput.raw(ec.1018935.tmax$MAX_TEMP,
ec.1018935.tmin$MIN_TEMP, ec.1018935.prec$ONE_DAY_PRECIPITATION,
tmax.dates, tmin.dates, prec.dates, base.range=c(1971, 2000))
```

---

climdexInput.csv *Method for creating climdexInput object from CSV files*

Description

This function creates a climdexInput object from data in CSV files.

Usage

```
climdexInput.csv(tmax.file = NULL, tmin.file = NULL, prec.file = NULL,
  data.columns = list(tmin = "tmin", tmax = "tmax", prec = "prec"),
  base.range = c(1961, 1990), na.strings = NULL, cal = "gregorian",
  date.types = NULL, n = 5, northern.hemisphere = TRUE,
  tavg.file = NULL, quantiles = NULL, temp.qtiles = c(0.1, 0.9),
  prec.qtiles = c(0.95, 0.99), max.missing.days = c(annual = 15, monthly =
  3), min.base.data.fraction.present = 0.1)
```

Arguments

`tmax.file`	Name of file containing daily maximum temperature data.
`tmin.file`	Name of file containing daily minimum temperature data.
`prec.file`	Name of file containing daily total precipitation data.
`data.columns`	Column names for tmin, tmax, and prec data.
`base.range`	Years to use for the baseline.
`na.strings`	Strings used for NA values; passed to `read.csv`.
`cal`	The calendar type used in the input files.
`date.types`	Column names for tmin, tmax, and prec data (see notes).
`n`	Number of days to use as window for daily quantiles.
`northern.hemisphere`	Whether this point is in the northern hemisphere.
`tavg.file`	Name of file containing daily mean temperature data.
`quantiles`	Threshold quantiles for supplied variables.
`temp.qtiles`	Quantiles to calculate for temperature variables
`prec.qtiles`	Quantiles to calculate for precipitation
`max.missing.days`	Vector containing thresholds for number of days allowed missing per year (annual) and per month (monthly).
`min.base.data.fraction.present`	Minimum fraction of base data that must be present for quantile to be calculated for a particular day

Details

This function takes input climate data in CSV files at daily resolution, and produces as output a ClimdexInput data structure. This data structure can then be passed to any of the routines used to compute the Climdex indices. The indices themselves are specified on the webpage cited in the references section.

Any of tmin.file (daily minimum temperature), tmax.file (daily maximum temperature), tavg.file (daily mean temperature), and prec.file (daily precipitation) can be passed in. tavg will be derived from the mean of tmax and tmin if it is not supplied. If any of tmin.file, tmax.file, and prec.file are not supplied, the set of indices which can be calculated will be limited to indices which do not involve the missing variables.

The `tmax.file`, `tmin.file`, and `prec.file` arguments should be names of CSV files containing dates and the data on which the indices are to be computed. The units are assumed to be degrees C for temperature, and mm/day for precipitation.

The `data.columns` argument is a vector consisting of named items tmax, tmin, and prec. These named items are used as the column names in their respective files when loading in CSV.

The `cal` argument is a textual description of the calendar type, as described in the documentation for `as.PCICt`.

The `date.types` argument is a list of lists containing two named items: `fields`, and `format`. The `fields` item is a vector of names consisting of the columns to be concatenated together with spaces. The `format` item is a date format as taken by `strptime`.

For more details on arguments, see `climdexInput.raw`.

Value

An object of class `climdexInput-class` for use with other climdex methods.

Note

Units are assumed to be mm/day for precipitation and degrees Celsius for temperature. No units conversion is performed internally.

References

http://etccdi.pacificclimate.org/list_27_indices.shtml

See Also

`climdex.pcic-package`, `climdexInput.raw`.

Examples

```
## This would create a climdexInput object from a set of filenames (already
## stored as variables), with a different date format.
## Not run: ci.csv <- climdexInput.csv(tmax.filename, tmin.filename,
prec.filename, date.types=list(list(fields=c("date"), format="%Y-%m-%d")))
## End(Not run)
```

---

climdexInput.raw *Method for creating climdexInput object from vectors of data*

Description

This function creates a climdexInput object from data already ingested into R.

Usage

```
climdexInput.raw(tmax = NULL, tmin = NULL, prec = NULL,
  tmax.dates = NULL, tmin.dates = NULL, prec.dates = NULL,
  base.range = c(1961, 1990), n = 5, northern.hemisphere = TRUE,
  tavg = NULL, tavg.dates = NULL, quantiles = NULL, temp.qtiles = c(0.1,
  0.9), prec.qtiles = c(0.95, 0.99), max.missing.days = c(annual = 15,
  monthly = 3), min.base.data.fraction.present = 0.1)
```

Arguments

tmax	Daily maximum temperature data.
tmin	Daily minimum temperature data.
prec	Daily total precipitation data.
tmax.dates	Dates for the daily maximum temperature data.
tmin.dates	Dates for the daily minimum temperature data.
prec.dates	Dates for the daily total precipitation data.
base.range	Years to use for the baseline.
n	Number of days to use as window for daily quantiles.
northern.hemisphere	Whether this point is in the northern hemisphere.
tavg	Daily mean temperature data.
tavg.dates	Dates for the daily mean temperature data.
quantiles	Threshold quantiles for supplied variables.
temp.qtiles	Quantiles to calculate for temperature variables
prec.qtiles	Quantiles to calculate for precipitation
max.missing.days	Vector containing thresholds for number of days allowed missing per year (annual) and per month (monthly).
min.base.data.fraction.present	Minimum fraction of base data that must be present for quantile to be calculated for a particular day

Details

Any of tmin (daily minimum temperature), tmax (daily maximum temperature), tavg (daily mean temperature), and prec (daily precipitation) can be passed in. tavg will be derived from the mean of tmax and tmin if it is not supplied. If any of tmin, tmax, and prec are not supplied, the set of indices which can be calculated will be limited to indices which do not involve the missing variables.

For all data supplied, the associated dates must also be supplied.

This function takes input climate data at daily resolution, and produces as output a ClimdexInput data structure. This data structure can then be passed to any of the routines used to compute the Climdex indices. The indices themselves are specified on the webpage cited in the references section. The base.range argument is a pair of 4 digit years which bound the data on which the base percentiles are calculated.

The tmax, tmin, and prec arguments are numeric vectors containing the data on which the indices are to be computed. The units are assumed to be degrees C for temperature, and mm/day for precipitation.

The tmax.dates, tmin.dates, and prec.dates arguments are vectors of type PCICt.

The n argument specifies the size of the window used when computing the percentiles used in climdex.tx10p, climdex.tn10p, climdex.tx90p, and climdex.tn90p.

The northern.hemisphere argument specifies whether the data came from the northern hemisphere. If FALSE, data is assumed to have come from the southern hemisphere. This is used when computing growing season length; if the data is from the southern hemisphere, growing season length is the growing season starting in the beginning of July of the year indicated, running to the end of June of the following year.

The quantiles argument allows the user to supply pre-computed quantiles. This is a list consisting of quantiles for each variable.

For each temperature variable, there are separate lists of quantiles for inbase and outbase, with these names. In both cases, quantiles within these lists are named q10 for the 10th percentile and q90 for the 90th percentile. Other percentiles would be named qnn for the nnth percentile. For the outbase quantiles, each element in the list is a vector of length 365 (or 360 in the case of 360-day calendars), corresponding to one value for each day of the year. For the inbase quantiles, each element in the list is an array of dimensions [365 or 360, nyr, nyr - 1], where nyr is the number of years in the base period. Each value corresponds to a quantile for each day, for each year, with a particular year replaced.

For precipitation variables, there is a named vector of quantiles, consisting of at least q95 and q99.

The temp.qtiles and prec.qtiles arguments allow the user to modify the quantiles calculated. For example, specifying temp.qtiles=c(0.10, 0.50, 0.90) would calculate the 10th, 50th, and 90th percentiles for temperature.

The min.base.fraction.present argument specifies the minimum fraction of data which must be present for a quantile to be calculated for a particular day. If the fraction of data present is less than this threshold, the quantile for that day will be set to NA.

The max.missing.days argument is a vector consisting of 'annual' (the number of days that can be missing in a year) and 'monthly' (the number of days that can be missing in a month. If one month in a year fails the test, the corresponding year will be omitted.

Value

An object of class climdexInput-class for use with other climdex methods.

Note

Units are assumed to be mm/day for precipitation and degrees Celsius for temperature. No units conversion is performed internally.

References

http://etccdi.pacificclimate.org/list_27_indices.shtml

See Also

climdex.pcic-package, strptime.

Examples

```
library(PCICt)

## Create a climdexInput object from some data already loaded in and
## ready to go.

## Parse the dates into PCICt.
tmax.dates <- as.PCICt(do.call(paste, ec.1018935.tmax[,c("year",
"jday")]), format="%Y %j", cal="gregorian")
tmin.dates <- as.PCICt(do.call(paste, ec.1018935.tmin[,c("year",
"jday")]), format="%Y %j", cal="gregorian")
prec.dates <- as.PCICt(do.call(paste, ec.1018935.prec[,c("year",
"jday")]), format="%Y %j", cal="gregorian")

## Load the data in.
ci <- climdexInput.raw(ec.1018935.tmax$MAX_TEMP,
ec.1018935.tmin$MIN_TEMP, ec.1018935.prec$ONE_DAY_PRECIPITATION,
tmax.dates, tmin.dates, prec.dates, base.range=c(1971, 2000))
```

---

ec.1018935 *EC example data*

Description

This is the Environment Canada CDCD (Canadian Daily Climate Data) precipitation, maximum temperature, and minimum temperature data for station 1018935 - William Head, BC, Canada.

Format

A data frame consisting of year, day of year, precipitation or daily maximum temperature or daily minimum temperature on that day, and observation flags.

Details

This is the Environment Canada CDCD (Canadian Daily Climate Data) precipitation, daily maximum temperature, and daily minimum temperature data for station 1018935 - William Head, BC, Canada. This is provided as example data for running the Climdex package.

See Also

climdexInput.raw, http://www.climate.weatheroffice.gc.ca/prods_servs/index_e.html#cdcd .

get.last.monthday.of.year

Get the last month and day of the year

Description

Get the last month and day of the year as a character sting, separated by the specified separator.

Usage

```
get.last.monthday.of.year(d, sep = "-")
```

Arguments

d	An exemplar date.
sep	Separator to use.

Details

This is a utility function necessitated by 360-day calendars. Works on PCICt objects.

Value

A string (like "12-30", or "12-31")

Examples

```
library(PCICt)
last.mday <- get.last.monthday.of.year(as.PCICt("2011-01-01", cal="360"))
```

get.outofbase.quantiles

Method for getting threshold quantiles for use in computing indices

Description

This function creates threshold quantiles for use with climdexInput.raw or climdexInput.csv.

Usage

```
get.outofbase.quantiles(tmax = NULL, tmin = NULL, prec = NULL,
  tmax.dates = NULL, tmin.dates = NULL, prec.dates = NULL,
  base.range = c(1961, 1990), n = 5, temp.qtiles = c(0.1, 0.9),
  prec.qtiles = c(0.95, 0.99), min.base.data.fraction.present = 0.1)
```

Arguments

tmax	Daily maximum temperature data.
tmin	Daily minimum temperature data.
prec	Daily total precipitation data.
tmax.dates	Dates for the daily maximum temperature data.
tmin.dates	Dates for the daily minimum temperature data.
prec.dates	Dates for the daily total precipitation data.
base.range	Years to use for the baseline.
n	Number of days to use as window for daily quantiles.
temp.qtiles	Quantiles to calculate for temperature variables
prec.qtiles	Quantiles to calculate for precipitation
min.base.data.fraction.present	Minimum fraction of base data that must be present for quantile to be calculated for a particular day
quantiles	Threshold quantiles for supplied variables.

Details

This function takes input climate data at daily resolution, and produces as output a set of threshold quantiles. This data structure can then be passed to climdexInput.raw or climdexInput.csv.

For more details on arguments, see climdexInput.raw.

Value

A set of threshold quantiles

Note

Units are assumed to be mm/day for precipitation and degrees Celsius for temperature. No units conversion is performed internally.

References

http://etccdi.pacificclimate.org/list_27_indices.shtml

See Also

climdex.pcic-package, climdexInput.raw.

Examples

```
library(PCICt)

## Create a climdexInput object from some data already loaded in and
## ready to go.

## Parse the dates into PCICt.
tmax.dates <- as.PCICt(do.call(paste, ec.1018935.tmax[,c("year",
"jday")]), format="%Y %j", cal="gregorian")
tmin.dates <- as.PCICt(do.call(paste, ec.1018935.tmin[,c("year",
"jday")]), format="%Y %j", cal="gregorian")
prec.dates <- as.PCICt(do.call(paste, ec.1018935.prec[,c("year",
"jday")]), format="%Y %j", cal="gregorian")

## Load the data in.
quantiles <- get.outofbase.quantiles(ec.1018935.tmax$MAX_TEMP,
ec.1018935.tmin$MIN_TEMP, ec.1018935.prec$ONE_DAY_PRECIPITATION,
tmax.dates, tmin.dates, prec.dates, base.range=c(1971, 2000))
```

---

get.series.lengths.at.ends

Get series length at ends

Description

This function takes a series of boolean values and returns a list of integers of the same length corresponding to the lengths at the ends of sequences of TRUE values.

Usage

```
get.series.lengths.at.ends(x, na.value = FALSE)
```

Arguments

x	Sequence of booleans.
na.value	Value to replace NAs with.

Details

It can often be useful to know how long a series of boolean values is. This function provides a method of knowing where and how long such sequences are.

Value

A vector consisting of the lengths of sequences of TRUE values at the location of the last TRUE value in the sequence, and zeroes elsewhere.

Examples

```
## Get lengths of sequences of TRUE values in a sequence
series.lengths <- get.series.lengths.at.ends(c(TRUE, TRUE, TRUE, FALSE,
TRUE, FALSE, TRUE, TRUE, TRUE, TRUE, FALSE))
```

`growing.season.length` *Flexible GSL function*

Description

This function computes the growing season length (GSL) given the input, which is allowed to vary considerably from the ETCCDI definitions.

Usage

```
growing.season.length(daily.mean.temp, date.factor, dates, northern.hemisphere,
  min.length = 6, t.thresh = 5, gsl.mode = c("GSL", "GSL_first",
  "GSL_max", "GSL_sum"))
```

Arguments

`daily.mean.temp`	Timeseries of daily mean temperature (in degrees C), padded out to end on a year boundary (ie: starts on January 1st of some year, ends on December 31st).
`date.factor`	Factor of the same length as daily.mean.temp that divides the timeseries up into years of data.
`dates`	The corresponding series of dates.
`northern.hemisphere`	Whether the data is from the northern hemisphere.
`min.length`	The minimum number of days above or below the threshold temperature that defines the start or end of a growing season.
`t.thresh`	The temperature threshold for being considered part of a growing season (in degrees C).
`gsl.mode`	The growing season length mode (ETCCDI mode is "GSL").

Details

This function is the function used to implement `climdex.gsl`. It's designed to be flexible to allow for experimentation and testing of new thresholds and methods.

If you need to use this code for experimentation in the southern hemisphere, you'll need to rip off the climdex.gsl code to rotate the year around so that July 1st is treated as January 1st.

See `climdex.gsl` for more information on what `gsl.mode` does.

Value

A vector containing the number of days in the growing season for each year.

See Also

climdex.gsl, climdexInput.csv.

Examples

```
library(PCICt)

## Create a climdexInput object from some data already loaded in and
## ready to go.

## Parse the dates into PCICt.
tmax.dates <- as.PCICt(do.call(paste, ec.1018935.tmax[,c("year",
"jday")]), format="%Y %j", cal="gregorian")
tmin.dates <- as.PCICt(do.call(paste, ec.1018935.tmin[,c("year",
"jday")]), format="%Y %j", cal="gregorian")
prec.dates <- as.PCICt(do.call(paste, ec.1018935.prec[,c("year",
"jday")]), format="%Y %j", cal="gregorian")

## Load the data in.
ci <- climdexInput.raw(ec.1018935.tmax$MAX_TEMP,
ec.1018935.tmin$MIN_TEMP, ec.1018935.prec$ONE_DAY_PRECIPITATION,
tmax.dates, tmin.dates, prec.dates, base.range=c(1971, 2000))

## Create an annual timeseries of the growing season length in days.
gsl <- growing.season.length(ci@data$tavg, ci@date.factors$annual, ci@dates,
                             ci@northern.hemisphere, gsl.mode="GSL") *
       ci@namasks$annual$tavg

## Print these out for testing purposes.
gsl
```

---

nday.consec.prec.max *Number of days (less than, greater than, etc) a threshold*

Description

Produces sums of values that exceed (or are below) the specified threshold.

Usage

```
nday.consec.prec.max(daily.prec, date.factor, ndays,
  center.mean.on.last.day = FALSE)
```

Arguments

daily.prec	Daily timeseries of precipitation.
date.factor	Factor to aggregate by.
ndays	Number of days in the running window.

center.mean.on.last.day
: Whether to center the n-day running mean on the last day of the series, instead of the middle day.

Details

This function takes a data series, the number of days in the running window, a date factor to aggregate by, and an optional modifier parameter (center.mean.on.last.day). It computes the n-day running sum of precipitation and returns the maximum n-day total precipitation per unit time, as defined by `date.factor`.

Value

A vector consisting of the maximum n-day sum of precipitation per time interval.

Examples

```
library(PCICt)

## Parse the dates into PCICt.
tmax.dates <- as.PCICt(do.call(paste, ec.1018935.tmax[,c("year",
"jday")]), format="%Y %j", cal="gregorian")
tmin.dates <- as.PCICt(do.call(paste, ec.1018935.tmin[,c("year",
"jday")]), format="%Y %j", cal="gregorian")
prec.dates <- as.PCICt(do.call(paste, ec.1018935.prec[,c("year",
"jday")]), format="%Y %j", cal="gregorian")

## Load the data in.
ci <- climdexInput.raw(ec.1018935.tmax$MAX_TEMP,
ec.1018935.tmin$MIN_TEMP, ec.1018935.prec$ONE_DAY_PRECIPITATION,
tmax.dates, tmin.dates, prec.dates, base.range=c(1971, 2000))

## Compute rx5day on a monthly basis.
rx5day <- nday.consec.prec.max(ci@data$prec, ci@date.factors$monthly, 5)
```

---

number.days.op.threshold

Number of days (less than, greater than, etc) a threshold

Description

Produces sums of values that exceed (or are below) the specified threshold.

Usage

```
number.days.op.threshold(temp, date.factor, threshold, op = "<")
```

Arguments

temp	Sequence temperature values.
date.factor	Factor to aggregate by.
threshold	Threshold to use.
op	Operator to use for comparison.

Details

This function takes a data series, a threshold, an operator, and a factor to aggregate by. It uses the operator to compare the threshold to the data series, creating a series of booleans, then sums the booleans according to the factor.

Value

A vector consisting of the number of values that meet the criteria in the given time period (as specified by date.factor).

Examples

```
library(PCICt)

## Parse the dates into PCICt.
tmax.dates <- as.PCICt(do.call(paste, ec.1018935.tmax[,c("year",
"jday")]), format="%Y %j", cal="gregorian")
tmin.dates <- as.PCICt(do.call(paste, ec.1018935.tmin[,c("year",
"jday")]), format="%Y %j", cal="gregorian")
prec.dates <- as.PCICt(do.call(paste, ec.1018935.prec[,c("year",
"jday")]), format="%Y %j", cal="gregorian")

## Load the data in.
ci <- climdexInput.raw(ec.1018935.tmax$MAX_TEMP,
ec.1018935.tmin$MIN_TEMP, ec.1018935.prec$ONE_DAY_PRECIPITATION,
tmax.dates, tmin.dates, prec.dates, base.range=c(1971, 2000))

## Calculate frost days.
fd <- number.days.op.threshold(ci@data$tmin,
                               ci@date.factors$annual, 0, "<")
```

---

percent.days.op.threshold

Lengths of strings of TRUE values

Description

Computes fraction of days above or below the baseline threshold for each day, and averages them using the date factor passed in.

Usage

```
percent.days.op.threshold(temp, dates, jdays, date.factor,
  threshold.outside.base, base.thresholds, base.range, op = "<",
  max.missing.days)
```

Arguments

temp	Sequence of temperature values.
dates	Sequence of associated dates.
jdays	Sequence of associated days of year.
date.factor	Factor to aggregate data using.
threshold.outside.base	Sequence of thresholds to be used for data outside the base period.
base.thresholds	Data structure containing sets of thresholds to be used inside the base period; see climdexInput-class.
base.range	Date range (type PCICt) of the baseline period.
op	Comparison operator to use.
max.missing.days	Maximum number of NA values per time period.

Details

This function computes fractions of days above or below baseline thresholds for each day, then aggregates them using `date.factor`. It is used to implement TN/TX 10/90p.

Value

A vector consisting of the mean fraction of days above or below the supplied set of thresholds.

Note

If date.factor is omitted, daily series will be returned.

See Also

climdexInput-class.

Examples

```
library(PCICt)

## Parse the dates into PCICt.
tmax.dates <- as.PCICt(do.call(paste, ec.1018935.tmax[,c("year",
"jday")]), format="%Y %j", cal="gregorian")
tmin.dates <- as.PCICt(do.call(paste, ec.1018935.tmin[,c("year",
"jday")]), format="%Y %j", cal="gregorian")
prec.dates <- as.PCICt(do.call(paste, ec.1018935.prec[,c("year",
```

```
"jday")]), format="%Y %j", cal="gregorian")

## Load the data in.
ci <- climdexInput.raw(ec.1018935.tmax$MAX_TEMP,
ec.1018935.tmin$MIN_TEMP, ec.1018935.prec$ONE_DAY_PRECIPITATION,
tmax.dates, tmin.dates, prec.dates, base.range=c(1971, 2000))

## Compute monthly tx90p.
tx90p <- percent.days.op.threshold(ci@data$tmax, ci@dates, ci@jdays,
                                   ci@date.factors$monthly,
                                   ci@quantiles$tmax$outbase$q90,
                                   ci@quantiles$tmax$inbase$q90,
                                   ci@base.range, ">",
                                   ci@max.missing.days['monthly']) *
         ci@namasks$monthly$tmax
```

`select.blocks.gt.length` *Select blocks of TRUE values of sufficient length.*

Description

Produces a sequence of booleans of the same length as input, with sequences of TRUE values shorter than n replaced with FALSE.

Usage

```
select.blocks.gt.length(d, n, na.value = FALSE)
```

Arguments

`d`	Sequence of booleans.
`n`	Longest sequence of TRUE to replace with FALSE.
`na.value`	Values to replace NAs with.

Details

This function takes a series of booleans and returns a sequence of booleans of equal length, with all sequences of TRUE of length `n` or shorter replaced with sequences of FALSE. NA values are replaced with `na.value`.

Value

A vector of booleans, with the length `n` or less sequences of TRUE replaced with FALSE.

Examples

```
## Return only the first sequence of TRUE... second sequence will be FALSE.
foo <- select.blocks.gt.length(c(rep(TRUE, 4), FALSE, rep(TRUE, 3)), 3)
```

simple.precipitation.intensity.index *Simple Precipitation Intensity Index*

Description

This function implements the ETCCDI Simple Precipitation Intensity Index.

Usage

```
simple.precipitation.intensity.index(daily.prec, date.factor)
```

Arguments

daily.prec	Data to compute index on.
date.factor	Date factor to split by.

Details

The simple precipitation intensity index is computed by taking the sum of precipitation in wet days (days with >1mm of precipitation), and dividing that by the number of wet days in the period. This gives the mean precipitation in wet days.

Value

The mean precipitation in wet days for each period (as defined by date.factor).

Examples

```
prec.dat <- c(0.1, 3.0, 4.3, 0.9, 1.3, 6.0, 0, 0, 4.0, 1)
phony.date.factor <- factor(rep(1:2, each=5))
sdii <- simple.precipitation.intensity.index(prec.dat, phony.date.factor)
```

spell.length.max *Maximum spell length*

Description

This function returns the longest string of days which exceed or are below the given threshold.

Usage

```
spell.length.max(daily.prec, date.factor, threshold, op, spells.can.span.years)
```

Arguments

daily.prec	Data to compute index on.
date.factor	Date factor to split by.
threshold	The threshold to compare to.
op	The operator to use to compare data to threshold.
spells.can.span.years	Whether spells can span years.

Details

This routine compares data to the threshold using the given operator, generating a series of TRUE or FALSE values. It then computes the lengths of sequences of TRUE values (spells) and chooses the longest spell in each period (as defined by date.factor).

The spells.can.span.years option controls whether spells must always terminate at the end of a period, or whether they may continue until the criteria ceases to be met or the end of the data is reached. The default for fclimdex is TRUE.

Value

A timeseries of maximum spell lengths for each period.

See Also

climdex.cdd.

Examples

```
prec.dat <- c(0.1, 3.0, 4.3, 1.9, 1.3, 6.0, 0, 0, 4.0, 1)
phony.date.factor <- factor(rep(1:2, each=5))

## With spells spanning years...
cwd <- spell.length.max(prec.dat, phony.date.factor, 1, ">=", TRUE)

## Without spells spanning years...
altcwd <- spell.length.max(prec.dat, phony.date.factor, 1, ">=", FALSE)
```

---

threshold.exceedance.duration.index

Sum of spell lengths exceeding daily threshold

Description

This function returns the number of spells of more than min.length days which exceed or are below the given threshold.

Usage

```
threshold.exceedance.duration.index(daily.temp, date.factor, jdays, thresholds,
  op = ">", min.length = 6, spells.can.span.years = TRUE,
  max.missing.days)
```

Arguments

daily.temp	Data to compute index on.
date.factor	Date factor to split by.
jdays	Timeseries of days of year.
thresholds	The thresholds to compare to.
op	The operator to use to compare data to threshold.
min.length	The minimum spell length to be considered.
spells.can.span.years	Whether spells can span years.
max.missing.days	Maximum number of NA values per time period.

Details

This routine compares data to the thresholds using the given operator, generating a series of TRUE or FALSE values; these values are then filtered to remove any sequences of less than `min.length` days of TRUE values. It then computes the lengths of the remaining sequences of TRUE values (spells) and sums their lengths.

The `spells.can.span.years` option controls whether spells must always terminate at the end of a period, or whether they may continue until the criteria ceases to be met or the end of the data is reached. The default for fclimdex is FALSE.

Value

A timeseries of maximum spell lengths for each period.

See Also

climdex.wsdi.

Examples

```
prec.dat <- c(0.1, 3.0, 4.3, 1.9, 1.3, 6.0, 0, 0, 4.0, 1)
phony.date.factor <- factor(rep(1:2, each=5))

## With spells spanning years...
alttedi <- threshold.exceedance.duration.index(prec.dat,
phony.date.factor, rep(1:5, 2), rep(1, 5), ">=", 2, TRUE, 1)

## Without spells spanning years...
tedi <- threshold.exceedance.duration.index(prec.dat, phony.date.factor,
rep(1:5, 2), rep(1, 5), ">=", 2, FALSE, 1)
```

`total.precip.op.threshold` *Sum of precipitation above a threshold*

Description

This function returns the sum of values above a threshold for each period (as defined by date.factor).

Usage

```
total.precip.op.threshold(daily.prec, date.factor, threshold, op)
```

Arguments

`daily.prec`	Data to compute index on.
`date.factor`	Date factor to split by.
`threshold`	The threshold to compare to.
`op`	The operator to use to compare data to threshold.

Details

This routine sums up all values which exceed or are below (depending on op) the given threshold.

Value

A timeseries of sums of numbers above the threshold for each period.

See Also

`climdex.r99ptot`.

Examples

```
prec.dat <- c(0.1, 3.0, 4.3, 1.9, 1.3, 6.0, 0, 0, 4.0, 1)
phony.date.factor <- factor(rep(1:2, each=5))

## Compute equiv of PRCPTOT
prec.sum <- total.precip.op.threshold(prec.dat, phony.date.factor, 1, ">=")
```

Index