Package 'agroclim'

April 8, 2022

Type Package

Title Climatic Indices for Agriculture

Version 0.2.0

Description Collection of functions to compute agroclimatic indices useful to zoning areas based on climatic variables and to evaluate the importance of temperature and precipitation for individual crops, or in general for agricultural lands.

Depends R (>= 3.5.0)

License GPL (>= 2)

Encoding UTF-8

LazyData true

RoxygenNote 7.1.2

Imports stats, ggplot2, gtable, plyr, zoo, abind, raster, ggpubr, cowplot, reshape, ggforce, ncdf4, easyNCDF, sp, multiApply, tools

NeedsCompilation no

Author Roberto Serrano-Notivoli [aut, cre] (<https://orcid.org/0000-0001-7663-1202>)

Maintainer Roberto Serrano-Notivoli <roberto.serrano@uam.es>

Repository CRAN

Date/Publication 2022-04-08 08:22:36 UTC

R topics documented:

`bedd` *Biologically effective degree-days (BEDD)*

Description

Biologically effective degree-days (BEDD). Gladstones, J. (1992).

Usage

```
bedd(mx, mn, lati, dates)
```

Arguments

`mx`	vector of daily maximum temperature series.
`mn`	vector of daily minimum temperature series.
`lati`	numeric value indicating the latitude of location.
`dates`	vector of dates corresponding with daily temprature series

Details

Depending on the latitude, the function detects the hemisphere and considers growing season from 1st April to 31st October (northern hemisphere) or from 1st October to 30rd April (southern hemisphere).

Value

The sum of degree-days (BEDD) is returned as a numeric value.

References

Gladstones, J. (1992) Viticulture and environment (Winetitles: Adelaide).

Examples

```
bedd(mx = daily_tmax,
  mn = daily_tmin,
  dates = seq.Date(as.Date('1981-01-01'),
          as.Date('2010-12-31'), by ='day'),
          lati = 42)
```

`calcGrid` *Calculation of indices over gridded datasets*

Description

Calculates any of the agroclimatic indices over a multidimensional dataset

Usage

```
calcGrid(
  mxnc = NULL,
  mnnc = NULL,
  varnametx = NULL,
  varnametn = NULL,
  fun,
  inienddays,
  dates,
  ncpu = 2,
  out = "test.nc",
  ...
)
```

Arguments

`mxnc`	a path to a NetCDF file(s) with maximum temperature data.
`mnnc`	a path to a NetCDF file(s) with minimum temperature data.
`varnametx`	a character string specifying the name of the variable to extract in maximum temperature NetCDF file.
`varnametn`	a character string specifying the name of the variable to extract in minimum temperature NetCDF file.
`fun`	name of the function to apply over the netcdf file.
`inienddays`	character vector indicating the initial and ending dates in which the input data must be splitted. Dates must be provided in format "YYY-MM-DD".

dates	vector of dates corresponding with daily temprature series.
ncpu	number of parallel processes to spawn for the use for parallel computation in multiple cores.
out	name of the NetCDF file where results will be saved (extension ".nc" must be included).
...	specific arguments matching the arguments of the target funcion.

Details

The "inienddays" argument only works if the input daily temperature ncdf file includes the time dimension in the format of "days since YYY-MM-DD". Otherwise, the function will return an error. All the arguments of the target function (function to be applied over the ncdf) must be stated ("...") or an unespecified error will appear. The function uses the argument "ncpu" to split the tasks into different cores taking advantage of the hardware.

Value

A NetCDF file is created in the path specified in the "out" parameter.

Examples

```
try(
calcGrid(mxnc = NULL, mnnc = 'tmin.nc', varnametn = 'tn', fun = 'frostProb',
         inienddays = NULL, dates = dates, ncpu = 4,
         out = 'fd_prob.nc', iniday = '07-01', endday = '06-30',
         type = 'doy', thres = 0, prob = 0.10)
)
```

---

coldMonth *Coldest month of the year*

Description

Calculates the mean temperature of the coldest month of the year

Usage

```
coldMonth(mn, dates, type = "temp", ...)
```

Arguments

mn	vector of daily minimum temperature series.
dates	vector of dates corresponding with daily temperature series
type	type of output. It can be "temp" for temperature or "month" for the number of the coldest month.
...	any other argument. It is expected to be "na.rm" in case NA values are required to be removed.

Value

Depending on argument type, the output will be a numeric value with the month (type = "month") or with temperature (type = "temp").

Examples

```
coldMonth(mn = daily_tmin,
          dates = seq.Date(as.Date('1981-01-01'),
           as.Date('2010-12-31'), by ='day'),
          type = 'temp',
          na.rm = TRUE)
```

`combine_plots` *Combine plots*

Description

This function combines two plots build with ggplot2 and use both legends.

Usage

```
combine_plots(plot1, plot2)
```

Arguments

`plot1`	First plot to draw.
`plot2`	Second plot that will be drawed over the first.

Value

A gtable object.

Examples

```
library(ggplot2)
df <- mtcars[, c("disp", "qsec", "cyl")]
plot1 <- ggplot(df, aes(x=cyl, y=qsec)) +
         geom_point() + theme_bw()
plot2 <- ggplot(df, aes(x=disp, y=qsec, color = factor(cyl))) +
         geom_line() + theme_void() + theme(legend.position = c(0.9, 0.8))
cb <- combine_plots(plot1, plot2)
plot(cb)
```

createNc *Creation of NetCDF from an array*

Description

Creates a NetCDF file from an array with values.

Usage

```
createNc(
  arr,
  varunit,
  varname,
  longname,
  lons,
  lats,
  inidate = NULL,
  scale = NULL,
  out
)
```

Arguments

arr	array containing the values to export in netcdf.
varunit	character. units in which variable is measured.
varname	character. name of the variable
longname	character. long name of the variable
lons	numeric. vector of longitudes.
lats	numeric. vector of latitudes.
inidate	date. initial day of observations.
scale	character. temporal units for the variable (usually "days", "months" or "years")
out	character. path to the output file.

Value

A NetCDF file is created in the path specified in the "out" parameter.

Examples

```
try(
createNc(arr = hi,
          varunit = 'ºC',
          varname = 'hi',
```

```
            longname = 'Huglin Index',
            lons = lon,
            lats = lat,
            inidate = '1980-01-01',
            scale = 'years',
            out = 'hi.nc')
    )
```

---

daily_tmax *Climatic dataset*

Description

Dataset with 30 years of daily maximum temperature.

Usage

```
daily_tmax
```

Format

numeric vector with maximum temperatures in celsius degrees.

`daily_tmax` numeric. daily maximum temperatures

Details

Daily maximum temperature

Source

Daily data source from the STEAD dataset of mainland Spain temperatures.

References

<http://dx.doi.org/10.20350/digitalCSIC/8622>

daily_tmin *Climatic dataset*

Description

Dataset with 30 years of daily minimum temperature.

Usage

```
daily_tmin
```

Format

numeric vectors with minimumtemperatures in celsius degrees.

`daily_tmin` numeric. daily minimum temperatures

Details

Daily minimum temperature

Source

Daily data source from the STEAD dataset of mainland Spain temperatures.

References

<http://dx.doi.org/10.20350/digitalCSIC/8622>

dataFromGrid *Extract data from a gridded dataset*

Description

This function extracts data from a gridded dataset based on given coordinates.

Usage

```
dataFromGrid(grid, varname, lon, lat, times = FALSE)
```

Arguments

`grid`	a RasterBrick or RasterStack object or a path to a NetCDF file.
`varname`	a character string specifying the name of the variable to extract in NetCDF file.
`lon`	numeric vector with longitude(s) of target location(s).
`lat`	numeric vector with latitude(s) of target location(s).
`times`	logical. If TRUE, time dimension will be extracted from data and the returned values will be in form of a zoo object. Only works for NetCDF and RasterBrick.

Details

The argument time searches the time dimension and its units in the gridded dataset (NetCDF or RasterBrick). If it can't be found, only the data will be returned.

Arguments lon and lat define the coordinates that must be in the same projection as the gridded dataset. If there is no spatial coincidence, returned object will be empty.

Value

A matrix with same columns as pairs of coordinates defined in lon and lat arguments. If times is TRUE, the matrix will be a zoo object.

Examples

```
try(
 dataFromGrid(grid = "tmax.nc",
                 varname = 'tn',
                 lon = c(41.39, 42),
                 lat = c(2.17, 2.5),
                 times = FALSE)
)
```

---

ehe *Extreme Heat Exposure (EHE)*

Description

Extreme Heat Exposure (EHE). Useful for climatic risks assessement on wheat and barley.

Usage

```
ehe(mx, dates, op = "first")
```

Arguments

mx	vector of daily maximum temperature series.
dates	vector of dates corresponding with daily temprature series
op	character. Indicates whether the output will be in date or numeric format.

Details

Adapted from Trnka et al. (2014). Event is triggered when the Tmax is above +35°C for at least three days during the period from five days after anthesis (supposed to be May-1st) to maturity (suposed to be July-31st). The minimum daily temperature is usually measured 2 m above ground; thus, the actual crop temperature might be even lower.

Value

If op = "first", the function returns the first day (date format) when the first event is triggered. If op ==’doy’, julian day is returned. If op = "number", the funciton returns the number of events occurred in the year.

References

Trnka M, Rotter RP, Ruiz-Ramos M, Kersebaum KC, Olesen JE, Zalud Z, Semenov MA (2014) Adverse weather conditions for European wheat production will become more frequent with climate change. Nature Climate Change volume 4, pages 637–643.

Examples

```
ehe(mx = daily_tmax,
dates = seq.Date(as.Date('1981-01-01'),
        as.Date('2010-12-31'), by ='day'),
op = 'first')
```

`firstFrost` *Mean first frost day*

Description

Calculates the first frost day within a predefined period.

Usage

```
firstFrost(
  mn,
  dates,
  iniday = "07-01",
  endday = "06-30",
  type = "doy",
  thres = 0
)
```

Arguments

`mn`	vector of daily minimum temperature series.
`dates`	vector of dates corresponding with daily temprature series
`iniday`	first day of the year ("dd-mm") when frost occurrence will be considered.
`endday`	last day of the year ("dd-mm") when frost occurrence will be considered.
`type`	type of output. It can be "doy" for day of the year (julian day) or "date" for data format ("dd-mm").
`thres`	temperature threshold considered to trigger frost occurrence (0 by default).

Details

The function is able to span over years. If the argument "enddate" is a date earlier than "inidate", the considered period will cover from "inidate" to the end of the year and from the beginning of the next year to "enddate".

Value

Depending on argument type, the output will be a numeric vector of julian days (type = "doy") or a vector of characters with dates (type = "date").

Examples

```
firstFrost(mn = daily_tmin,
            dates = seq.Date(as.Date('1981-01-01'),
                             as.Date('2010-12-31'), by ='day'),
            iniday = '07-01',
            endday = '06-30',
            type = 'doy')
```

`firstTemp` *Mean first day with a temperature over a threshold*

Description

Calculates the first day, within a predefined period, with a temperature over a threshold

Usage

```
firstTemp(
  mx,
  dates,
  iniday = "01-01",
  endday = "09-01",
  type = "doy",
  thres = 35
)
```

Arguments

`mx`	vector of daily (usually maximum) temperature series.
`dates`	vector of dates corresponding with daily temprature series
`iniday`	first day of the year ("dd-mm") when frost occurrence will be considered.
`endday`	last day of the year ("dd-mm") when frost occurrence will be considered.
`type`	type of output. It can be "doy" for day of the year (julian day) or "date" for data format ("dd-mm").
`thres`	temperature threshold considered to trigger occurrence (+35 by default).

Details

The function is able to span over years. If the argument "enddate" is a date earlier than "inidate", the considered period will cover from "inidate" to the end of the year and from the beginning of the next year to "enddate".

Value

Depending on argument type, the output will be a numeric vector of julian days (type = "doy") or a vector of characters with dates (type = "date").

Examples

```
firstTemp(mx = daily_tmax,
            dates = seq.Date(as.Date('1981-01-01'),
                             as.Date('2010-12-31'), by ='day'),
            iniday = '01-01',
            endday = '01-09',
            type = 'doy',
            thres = 35)
```

---

frostDays *Number of frost days*

Description

Calculates the number of frost days within a predefined period.

Usage

```
frostDays(mn, dates, iniday = "07-01", endday = "06-30", thres = 0)
```

Arguments

mn	vector of daily minimum temperature series.
dates	vector of dates corresponding with daily temprature series
iniday	first day of the year ("dd-mm") when frost occurrence will be considered.
endday	last day of the year ("dd-mm") when frost occurrence will be considered.
thres	temperature threshold considered to trigger frost occurrence (0 by default).

Details

Despite the logical threshold of temperature is 0 ºC to determine frost occurrence, the argument "thres" is open to change in case of different units of temperature.

Value

A numeric vector with the annual number of frost days is returned.

Examples

```
frostDays(mn = daily_tmin,
          dates = seq.Date(as.Date('1981-01-01'),
                           as.Date('2010-12-31'), by ='day'),
          iniday = '07-01',
          endday = '06-30')
```

---

frostFreqs *Frequency of frosts*

Description

This function calculates the frequency of frost occurrence, by month.

Usage

```
frostFreqs(
  mn,
  dates,
  thres = 0,
  out = NULL,
  iniday = "07-01",
  endday = "06-30"
)
```

Arguments

mn	vector containing the daily minimum temperature series.
dates	sequence of daily dates coinciding with temperature data series.
thres	temperature threshold considered to trigger frost occurrence (0 by default).
out	system path to the generated output file ('.pdf').
iniday	first day of the year ("dd-mm") to consider for first and last occurrence of frost.
endday	last day of the year ("dd-mm") to consider for first and last occurrence of frost.

Details

Despite the logical threshold of temperature is 0 ºC to determine frost occurrence, the argument "thres" is open to change in case of different units of temperature.

Value

If the output path is defined, a pdf file is created.

Examples

```
frostFreqs(mn = daily_tmin, dates = seq.Date(as.Date('1981-01-01'),
                              as.Date('2010-12-31'), by ='day'), thres = 0)
```

`frostProb` *First day in the year where P(tmin<0) <= prob*

Description

Calculates the first day in the year where the probability if frost is under a threshold.

Usage

```
frostProb(
  mn,
  dates,
  iniday = "07-01",
  endday = "06-30",
  type = "doy",
  thres = 0,
  prob = 0.1
)
```

Arguments

`mn`	vector of daily minimum temperature series.
`dates`	vector of dates corresponding with daily temprature series
`iniday`	first day of the year ("dd-mm") when frost occurrence will be considered.
`endday`	last day of the year ("dd-mm") when frost occurrence will be considered.
`type`	type of output. It can be "doy" for day of the year (julian day) or "date" for data format ("dd-mm").
`thres`	temperature threshold considered to trigger frost occurrence (0 by default).
`prob`	numeric. Threshold indicating the probability (0.10 by default)

Details

The function is able to span over years. If the argument "enddate" is a date earlier than "inidate", the considered period will cover from "inidate" to the end of the year and from the beginning of the next year to "enddate".#' @return depending on argument type, the output will be a numeric vector of julian days (type = "doy") or a vector of characters with dates (type = "date").

Value

A numeric vector with the annual number of frost probability is returned.

Examples

```
frostProb(mn = daily_tmin,
  dates = seq.Date(as.Date('1981-01-01'),
                           as.Date('2010-12-31'), by ='day'),
  iniday = '07-01',
  endday = '06-30',
  type = 'date',
  prob = 0.10)
```

---

gdd *Growing Degree Days*

Description

Growing Degree Day (GDD) or Winkler index. Useful as a zoning tool to differentiate between grape varieties and climate (Winkler et al. 1974).

Usage

```
gdd(mx, mn, dates, lati = NULL, iniday, endday)
```

Arguments

mx	vector of daily maximum temperature series.
mn	vector of daily minimum temperature series.
dates	vector of dates corresponding with daily temprature series
lati	(optional) numeric value indicating the latitude of location.
iniday	initial date in text format ("YYYY-mm-dd"). If "lati" is defined, "iniday" and "endday" are automatically set.
endday	last date in text format ("YYYY-mm-dd"). If "lati" is defined, "iniday" and "endday" are automatically set.

Details

Depending on the latitude, the function detects the hemisphere and considers growing season from 1st April to 31st October (northern hemisphere) or from 1st October to 30rd April (southern hemisphere).

Value

The sum of growing-degree-days is returned as a numeric value.

References

Winkler AJ, Cook JA, Kliwer WM, Lider LA (1974) General viticulture. University of California Press, Berkeley, CA

Examples

```
gdd(mx = daily_tmax,
  mn = daily_tmin,
  dates = seq.Date(as.Date('1981-01-01'),
          as.Date('2010-12-31'), by ='day'),
          lati = 42)
```

---

gst *Growing Season Temperature (GST)*

Description

Growing Season Temperature (GST). Mean daily temperature in growing season.

Usage

```
gst(mx, mn, lati, dates)
```

Arguments

mx	vector of daily maximum temperature series.
mn	vector of daily minimum temperature series.
lati	numeric value indicating the latitude of location.
dates	vector of dates corresponding with daily temprature series

Details

Depending on the latitude, the function detects the hemisphere and considers growing season from 1st April to 31st October (northern hemisphere) or from 1st October to 30rd April (southern hemisphere).

Value

A numeric vector with annual values is returned.

References

Jones G, Duff A, Hall A, Myers J (2010) Spatial Analysis of Climate in Winegrape Growing Regions in the Western United States. Am. J. Enol. Vitic. 61:3.

Examples

```
gst(mx = daily_tmax,
  mn = daily_tmin,
  dates = seq.Date(as.Date('1981-01-01'),
          as.Date('2010-12-31'), by ='day'),
          lati = 42)
```

---

hi *Huglin Heliothermal Index (HI)*

Description

Huglin Heliothermal Index (HI). Useful as a zoning tool (Huglin 1978).

Usage

```
hi(mx, mn, dates, lati)
```

Arguments

mx	vector of daily maximum temperature series.
mn	vector of daily minimum temperature series.
dates	vector of dates corresponding with daily temprature series
lati	numeric value indicating the latitude of location.

Details

Depending on the latitude, the function detects the hemisphere and considers growing season from 1st April to 30rd September (northern hemisphere) or from 1st October to 31st March (southern hemisphere).

Value

A numeric vector with annual values is returned.

References

Huglin P. (1978) Nouveau mode d'evaluation des possibilities heliothermiques d'un milieu viticole. CR Acad Agr 64: 1117–1126

Examples

```
hi(mx = daily_tmax,
  mn = daily_tmin,
  dates = seq.Date(as.Date('1981-01-01'),
          as.Date('2010-12-31'), by ='day'),
          lati = 42)
```

`lastFrost` *Mean last frost day*

Description

Calculates the last frost day within a predefined period.

Usage

```
lastFrost(
  mn,
  dates,
  iniday = "07-01",
  endday = "06-30",
  type = "doy",
  thres = 0
)
```

Arguments

`mn`	vector of daily minimum temperature series.
`dates`	vector of dates corresponding with daily temprature series
`iniday`	first day of the year ("dd-mm-YYYY") when frost occurrence will be considered.
`endday`	last day of the year ("dd-mm-YYYY") when frost occurrence will be considered.
`type`	type of output. It can be "doy" for day of the year (julian day) or "date" for data format ("dd-mm").
`thres`	temperature threshold considered to trigger frost occurrence (0 by default).

Details

The function is able to span over years. If the argument "enddate" is a date earlier than "inidate", the considered period will cover from "inidate" to the end of the year and from the beginning of the next year to "enddate".

Value

A numeric vector with annual values is returned.

depending on argument type, the output will be a numeric vector of julian days (type = "doy") or a vector of characters with dates (type = "date").

Examples

```
lastFrost(mn = daily_tmin,
  dates = seq.Date(as.Date('1981-01-01'),
                           as.Date('2010-12-31'), by ='day'),
  iniday = '07-01',
  endday = '06-30',
  type = 'date')
```

`pentadProbs` *Pentadic probabilities*

Description

This function calculates the probability od occurrence of a temperature range in a five-day row (pentad).

Usage

```
pentadProbs(
  mx,
  mn,
  temp_dates,
  out = NULL,
  min_abs = -20,
  max_abs = 44,
  plot_years = NULL,
  show_legend = TRUE
)
```

Arguments

`mx`	vector containing the daily maximum temperature series.
`mn`	same as mx but with daily minimum temperature series.
`temp_dates`	sequence of daily dates coinciding with temperature data series.
`out`	system path to the generated output file ('.pdf').
`min_abs`	minimum absolute temperature used as threshold to plot data.
`max_abs`	same as min_abs but with maximum absolute temperature.

plot_years	numeric vector indicating the frequencies of individual years to plot over the averages.
show_legend	logical value indicating wether legend will be shown or not.

Details

The numer of individual years to plot over the average frequencies is unlimited. However, it is recommended to add only a few due to the difficult of reading a lot of information in the same plot.

The colors of the individual years are random, so it is possible to get different colors for same years in two runs of same plot.

Value

If the output path is defined, a pdf file is created.

Examples

```
# basic plot with legend
 pentadProbs(mx = daily_tmax, mn = daily_tmin,
              temp_dates = seq.Date(as.Date('1981-01-01'),
              as.Date('2010-12-31'), by ='day'),
              out = NULL, min_abs = -20, max_abs = 44,
              plot_years = NULL, show_legend = TRUE)

# years 1981 and 2009 are plotted over the average frequencies
 pentadProbs(mx = daily_tmax, mn = daily_tmin,
              temp_dates = seq.Date(as.Date('1981-01-01'),
              as.Date('2010-12-31'), by ='day'),
              out = NULL, min_abs = -20, max_abs = 44,
              plot_years = c(1981, 2009), show_legend = TRUE)
```

---

tempDayprob *First day in the year where P(tmax>threshold) >= probs*

Description

Calculates the first day in the year where the probability of temperature over a threshold is higher than a predefined threshold.

Usage

```
tempDayprob(
  mx,
  dates,
  iniday = "01-01",
```

```
  endday = "09-01",
  type = "doy",
  thres = 35,
  prob = 0.1
)
```

Arguments

mx	vector of daily (usually maximum) temperature series.
dates	vector of dates corresponding with daily temprature series
iniday	first day of the year ("dd-mm-YYYY") when frost occurrence will be considered.
endday	last day of the year ("dd-mm-YYYY") when frost occurrence will be considered.
type	type of output. It can be "doy" for day of the year (julian day) or "date" for data format ("dd-mm").
thres	temperature threshold considered to trigger frost occurrence (0 by default).
prob	numeric. Threshold indicating the probability (0.10 by default)

Details

The function is able to span over years. If the argument "enddate" is a date earlier than "inidate", the considered period will cover from "inidate" to the end of the year and from the beginning of the next year to "enddate".#' @return depending on argument type, the output will be a numeric vector of julian days (type = "doy") or a vector of characters with dates (type = "date").

Value

A numeric vector with annual values is returned.

Examples

```
tempDayprob(mx = daily_tmax,
  dates = seq.Date(as.Date('1981-01-01'),
                           as.Date('2010-12-31'), by ='day'),
  iniday = '01-01',
  endday = '09-01',
  type = 'date',
  prob = 0.10)
```

tempProb *Probability of exceed a predefined temperature value*

Description

Calculates the first day in the year where the probability of temperature over a threshold is higher than a predefined threshold.

Usage

```
tempProb(mx, dates, thres = 20, month = NULL)
```

Arguments

mx	vector of daily (usually maximum) temperature series.
dates	vector of dates corresponding with daily temprature series
thres	temperature threshold considered to trigger occurrence.
month	integer between 1 and 12 indicating the month to compute the probability.

Value

A numeric vector with annual values is returned.

Examples

```
tempProb(mx = daily_tmax,
dates = seq.Date(as.Date('1981-01-01'),
                         as.Date('2010-12-31'), by ='day'),
thres = 20,
month = NULL)
```

warmMonth *Warmest month of the year*

Description

Calculates the mean temperature of the warmest month of the year

Usage

```
warmMonth(mx, dates, type = "temp", ...)
```

Arguments

mx	vector of daily maximum temperature series.
dates	vector of dates corresponding with daily temperature series
type	type of output. It can be "temp" for temperature or "month" for the number of the warmest month.
...	any other argument. It is expected to be "na.rm" in case NA values are required to be removed.

Value

Depending on argument type, the output will be a numeric value with the month (type = "month") or with temperature (type = "temp").

Examples

```
warmMonth(mx = daily_tmax,
          dates = seq.Date(as.Date('1981-01-01'),
           as.Date('2010-12-31'), by ='day'),
          type = 'temp',
          na.rm = TRUE)
```

Index