Package: weaana (via r-universe)

September 8, 2024

Maintainer Bangyou Zheng <bangyou.zheng@csiro.au>

Title Analysis the Weather Data

Type Package

Description Functions are collected to analyse weather data for agriculture purposes including to read weather records in multiple formats, calculate extreme climate index.

License MIT + file LICENSE

URL https://weaana.bangyou.me/, https://github.com/byzheng/weaana

BugReports https://github.com/byzheng/weaana/issues

Encoding UTF-8

Version 0.2.0

Date 2021-09-06

LazyData true

Depends R (>= 3.5.0)

Imports methods, stats, settings, reshape2, lubridate, magrittr, rlang, dplyr, tibble

RoxygenNote 7.1.2

Suggests rmarkdown, knitr, testthat

VignetteBuilder knitr

Repository https://byzheng.r-universe.dev

RemoteUrl https://github.com/byzheng/weaana

RemoteRef HEAD

RemoteSha 3f88b89712d8ffa6126b37a614f9b8c52f17e5cb

Contents

`changeWeatherRecords` *Change weather records*

Description

Change weather records

Change weather records

Usage

```
changeWeatherRecords(object, ...)

## S4 method for signature 'WeaAna'
changeWeatherRecords(object, ...)
```

Arguments

`object`	A WeaAna object.
`...`	New weather records

Value

A new WeaAna object with updated records

`climate_by_stages` *Summarise the climate variable by growth stages*

Description

Summarise the climate variable by growth stages

Usage

```
climate_by_stages(
  climates,
  sowing,
  emergence,
  heading = NULL,
  flowering = NULL,
  maturity,
  latitude
)
```

Arguments

`climates`	a data.frame for climate records
`sowing`	date. an vector of sowing date
`emergence`	numeric (days after sowing). an vector of emergence date
`heading`	numeric (days after sowing). an vector of heading date (optional. see details)
`flowering`	numeric (days after sowing). an vector of flowering time (optional. see details)
`maturity`	numeric (days after sowing). an vector of maturity time
`latitude`	latitude

Details

Define of growth stages

- S0: From start of year to emergence
- S1: From emergence to flowering time - 300Cd
- S2: From flowering time - 300Cd to flowering time + 100Cd
- S3: From flowering time + 100 Cd to flowering time + 600Cd
- S4: From flowering + 600Cd to maturity

Climate variables

- stage: defination of stages
- n: Number of days in each stage
- avgt: average temperature (C)
- sum.tt: total thermal time (Cd) with base temperature 0C
- avg.mint: average minimum temperature
- avg.maxt: average maximum temperature
- sum.rain: total rainfall
- avg.evap: average evapration
- avg.radn average radiation
- hot.days: number of hot days (daily maximum temperature is more than 30C)
- very.hot.days: number of very hot days (daily maximum temperature is more than 35C)
- frost.days: number of frost days (daily minimum temperature is less than 0C)
- hot.sum: total thermal time above 30C of hot days (daily maximum temperature is more than 30C)
- very.hot.sum: total thermal time above 35C of very hot days (daily maximum temperature is more than 35C)
- frost.sum: total thermal time below 0C of frost days (daily minimum temperature is less than 0C)
- vpd: vapour-pressure deficit
- te: transpiration efficiency
- bio.radn: bio.radn
- bio.water: bio.water
- bio.tt: bio.tt
- ptq: photothermal quotient
- avt.diffuse.radn: average diffuse radiation

Value

a data.frame for summarised climate variable by stages. See details for more information.

Examples

```
if (FALSE) {
   sowing <- rep(as.Date("1981-05-01"), 10)
   emergence <- rep(10, 10)
   heading <- NULL
   flowering <- runif(10) * 20 + 50
   maturity <- runif(10) * 20 + 100
   latitude <- -27
   res <- climate_by_stages(climates = climates,
                            sowing = sowing,
                            emergence = emergence,
                            heading = heading,
                            flowering = flowering,
                            maturity = maturity,
                            latitude = latitude)
}
```

convert2Records *Convert a data frame to weaana class*

Description

Convert a data frame to weaana class

Usage

```
convert2Records(infor, records)
```

Arguments

`infor`	A list or data frame of site information
`records`	A data frame will convert to records

Value

A new WeaAna object

createWeaAna *create WeaAna class*

Description

create WeaAna class

Usage

```
createWeaAna(mets)
```

Arguments

`mets`	A list contained information of weather records.

Value

A new WeaAna class

dayLength *The time elapsed in hours between the specified sun angle from 90 degree in am and pm. +ve above the horizon, -ve below the horizon.*

Description

The time elapsed in hours between the specified sun angle from 90 degree in am and pm. +ve above the horizon, -ve below the horizon.

Usage

```
dayLength(doy, lat, angle = -6)
```

Arguments

`doy`	day of year number
`lat`	latitude of site (deg)
`angle`	angle to measure time between, such as twilight (deg). angular distance between 90 deg and end of twilight - altitude of sun. +ve up, -ve down.

Value

day length in hours

diurnalT *Calculate the diurnal variation in air temperature with Parton and Logan, 1981*

Description

Calculate the diurnal variation in air temperature. Parton WJ, Logan JA (1981) A model for diurnal variation in soil and air temperature. Agricultural Meteorology, 23, 205?216. Codes copied from APSIM Utilities.cpp

Usage

```
diurnalT(maxt, mint, doy, hour, latitude, A = 1.5, B = 4, C = 1)
```

Arguments

`maxt`	maximum daily temperature
`mint`	minimum daily temperature
`doy`	day of year
`hour`	hour from 1 to 24
`latitude`	latitude in radials
`A`	is the time lag in temperature after noon
`B`	is coef that controls temperature decrease at night
`C`	is the time lag for min temperature after sunrise

Value

A vector with diurnal air temperature

Examples

```
diurnalT(maxt = 20, mint = 10, doy  = 1,
   hour = seq(from = 1, to = 23.99, by = 0.1),
   latitude = -10, A = 1.5, B = 4, C = 1)
```

`getWeatherRecords` *Get all weather records by year range*

Description

Get all weather records by year range

Get all weather records by year range

Usage

```
getWeatherRecords(object, ...)

## S4 method for signature 'WeaAna'
getWeatherRecords(object, yrange = NULL, vars = "all", ...)
```

Arguments

`object`	A WeaAna object.
`...`	Other arguments
`yrange`	Year range.
`vars`	Variable

Value

A data frame with all weather records

Examples

```
library(weaana)
data( "WeatherRecordsDemo" )
getWeatherRecords( records, yrange = c( 2008, 2009 ) )
getWeatherRecords( records, yrange = c( 2008, 2009 ), length = 10 )
```

`interpolationFunction` *Return a y value from a linear interpolation function*

Description

Return a y value from a linear interpolation function

Usage

```
interpolationFunction(x, y, values, split = "\\s+")
```

Arguments

`x`	x
`y`	y
`values`	values
`split`	split

Value

The interpolated values

`mov` *Calculate the moving values*

Description

Calculate the moving values

Usage

```
mov(x, k = 10, shift = "centre", fun = "mean")
```

Arguments

`x`	A vector to calculate moving values
`k`	The moving windows
`shift`	if shift = "centre", then values are shifted to centre. if shift = "begin", then values are at begin of period. if shift = "end", then values are at end of period. The default value (centre) will be used if shift is other value.
`fun`	The method to calculate moving values. Curruntly, only "mean", "max", "min", and "sum" are supported. A NULL will be returned for any other values

Value

The moving value of vector x at moving windows k. A NULL will be returned for any unsupported fun

---

`mov.avg` *Use Calculate the moving average. For compatibility only.*

Description

Note that for n = odd, can average at central period. If n = even, must average at end of period and then shift values

Usage

```
mov.avg(x, k = 10, shift = "centre")
```

Arguments

`x`	A vector to calculate moving average
`k`	The moving windows
`shift`	if shift = "centre", then values are shifted to centre. if shift = "begin", then values are at begin of period. if shift = "end", then values are at end of period. The default value (centre) will be used if shift is other value.

Value

The moving average of vector x at moving windows n

mov.max *Calculate the moving maximum. For compatibility only.*

Description

Calculate the moving maximum. For compatibility only.

Usage

```
mov.max(x, k, shift = "centre")
```

Arguments

`x` A vector to calculate moving maximum

`k` The moving windows

`shift` if shift = "centre", then values are shifted to centre. if shift = "begin", then values are at begin of period. if shift = "end", then values are at end of period. The default value (centre) will be used if shift is other value.

Value

The moving maximum of vector x at moving windows k

mov.min *Calculate the moving minimum. For compatibility only.*

Description

Calculate the moving minimum. For compatibility only.

Usage

```
mov.min(x, k, shift = "centre")
```

Arguments

`x` A vector to calculate moving minimum

`k` The moving windows

`shift` if shift = "centre", then values are shifted to centre. if shift = "begin", then values are at begin of period. if shift = "end", then values are at end of period. The default value (centre) will be used if shift is other value.

Value

The moving minimum of vector x at moving windows k

mov.sum *Calculate the moving sum. For compatibility only.*

Description

Calculate the moving sum. For compatibility only.

Usage

```
mov.sum(x, k, shift = "centre")
```

Arguments

x	A vector to calculate moving sum
k	The moving windows
shift	if shift = "centre", then values are shifted to centre. if shift = "begin", then values are at begin of period. if shift = "end", then values are at end of period. The default value (centre) will be used if shift is other value.

Value

The moving sum of vector x at moving windows k

readWeatherRecords *Read weather records from a file list and/or a folder list*

Description

Read weather records from a file list and/or a folder list

Usage

```
readWeatherRecords(
  dataFiles = NULL,
  dataFolders = NULL,
  dataFormat = "APSIM",
  dataWeather = NULL,
  load.later = FALSE,
  ...
)
```

Arguments

`dataFiles`	A character vector to specify the path of weather data files.
`dataFolders`	A character vector to specify the path of weather data folders.
`dataFormat`	The format of weather data file.
`dataWeather`	A data.frame for existing data.
`load.later`	Whether load weather records now or later. "dataFroamt" should be One of "APSIM" and "RDATA".
`...`	Other arguments

Value

A WeaAna class which contains all weather data.

`records` *Demo weather records*

Description

Demo weather records

Usage

```
records
```

Format

An object of class `WeaAna` of length 1.

`result-class` *Define the class for statistics results*

Description

Define the class for statistics results

Slots

`name` Name of result

`type` Type of result

show,WeaAna-method *Show basic information of class WeaAna*

Description

Show the name, number, latitude, longitude of all weather stations.

Usage

```
## S4 method for signature 'WeaAna'
show(object)
```

Arguments

`object`	WeaAna objects

Examples

```
library(weaana)
data( "WeatherRecordsDemo" )
show( records )
records
```

siteInfor *Get site information*

Description

Get site information

Get site information

Get site information

Usage

```
siteInfor(object, ...)

## S4 method for signature 'WeaAna'
siteInfor(object, load.now = FALSE)

## S4 method for signature 'WeaAnaSite'
siteInfor(object, load.now = FALSE)
```

Arguments

`object`	A WeaAnaSite object.
`...`	Not used
`load.now`	Whether load site information

Value

Site information in the WeaAna object

Site information in the WeaAnaSite object

Examples

```
library(weaana)
data( "WeatherRecordsDemo" )
siteInfor( records )
siteInfor( records, load.now = TRUE )
```

`sphericalDistance` *Calculate the sphere distance*

Description

Calculate the sphere distance

Usage

```
sphericalDistance(lat1, lon1, lat2, lon2)
```

Arguments

`lat1`	Latitude
`lon1`	Longitude
`lat2`	Latitude
`lon2`	Longitude

Value

Distance in km

thermalTime *Calculate thermal time using cardinal temperatures*

Description

Calculate thermal time using cardinal temperatures

Usage

```
thermalTime(weather, x_temp, y_temp, method = NULL)
```

Arguments

weather	WeaAna object
x_temp	The cardinal temperatures
y_temp	The effective thermal time
method	The method to calculate thermal time. The default method is (maxt + mint) / 2 - base. The three hour temperature methods will be used if method = '3hr'

Value

A data.frame with three columns: year, day and thermalTime.

Examples

```
met_file <- system.file("extdata/WeatherRecordsDemo1.met", package = "weaana")
records <- readWeatherRecords(met_file)
x_temp <- c(0, 26, 34)
y_temp <- c(0, 26, 0)
res <- thermalTime(records, x_temp, y_temp)
head(res)
res <- thermalTime(records, x_temp, y_temp, method = "3hr")
head(res)
```

thermalTimeDaily *Calculate thermal time using cardinal temperatures*

Description

Calculate thermal time using cardinal temperatures

Usage

```
thermalTimeDaily(mint, maxt, x_temp, y_temp, method = NULL)
```

Arguments

`mint`	The minimum temperature
`maxt`	The maximum temperature
`x_temp`	The cardinal temperatures
`y_temp`	The effective thermal time
`method`	The method to calculate thermal time. The default method is (maxt + mint) / 2 - base. The three hour temperature methods will be usesd if method = ’3hr’

Value

The thermal time.

Examples

```
mint <- c(0, 10)
maxt <- c(30, 40)
x_temp <- c(0, 20, 35)
y_temp <- c(0, 20, 0)
thermalTimeDaily(mint, maxt, x_temp, y_temp)
thermalTimeDaily(mint, maxt, x_temp, y_temp, method = '3hr')
```

`thermalTimeHourly` *Calculate thermal time using the hourly temperature (non daily temperature)*

Description

Calculate thermal time using the hourly temperature (non daily temperature)

Usage

```
thermalTimeHourly(timestamp, temperature, x_temp, y_temp)
```

Arguments

`timestamp`	The timestamp of weather records
`temperature`	The temperature
`x_temp`	The cardinal temperatures
`y_temp`	The effective thermal time

Value

A data frame with daily thermal time

Examples

```
met_file <- system.file("extdata/WeatherHourly.csv", package = "weaana")
hourly <- read.csv(met_file, as.is = TRUE)

hourly$timestamp <- as.POSIXct(hourly$timestamp, format = "%Y-%m-%dT%H:%M:%SZ")
x_temp <- c(0, 20, 35)
y_temp <- c(0, 20, 0)
thermalTimeHourly(hourly$timestamp, hourly$temperature, x_temp, y_temp)
```

ttest_ts *Significantly t-test with auto-correlation for time serial data*

Description

Method is presented by Santer et al. 2000

Usage

```
ttest_ts(y, slope = NULL)
```

Arguments

y	A vector of time serial data
slope	Whether export slope

Value

p values of t-test

wcal *Calculate weather variables through function or a string formula.*

Description

There are two modes to use wcal, function mode if FUN is not null, and string formula mode if FUN is NULL.

Usage

```
wcal(object, ...)

## S4 method for signature 'WeaAna'
wcal(object, FUN = NULL, ..., var.args = NULL, var.name = NULL)
```

Arguments

object	A WeaAna objects.
...	Optional arguments to FUN in function mode. String formulas if FUN is NULL.
FUN	A function to be used which results should have the same length as original records.
var.args	Arguments of weather variable pass to FUN.
var.name	Variable name is used if FUN is not NULL.

Examples

```
library(weaana)
data( "records" )
# Daily mean temperature
wcal( records, avgt2 = "( maxt + mint ) / 2" )
# Moving average temperature
wcal( records, FUN = mov.avg, var.args = "avgt", k = 5, shift = "begin", var.name = "mov.avg" )
```

WeaAna-class *Define the class for multiple sites*

Description

Define the class for multiple sites

Slots

- num total number of weather station
- records A pointer vector to weather records of each site
- result A pointer for all results name and type.

WeaAnaSite-class *Define the class of WeaAna*

Description

Define the class of WeaAna

Slots

`name` Name of weather station

`number` Station number of weather station

`latitude` Latitude of weather station

`longitude` Latitude of weather station

`tav` Annual average ambient temperature

`amp` Annual amplitude in mean monthly temperature

`marker` The extra marker for this site

`year` A vector of year of weather station

`day` A vector of day of weather station

`radn` A vector of radiation of weather station

`maxt` A vector of maximum temperature of weather station

`mint` A vector of minimum temperature of weather station

`evap` A vector of evaporation of weather station

`rain` A vector of rainfall of weather station

`vp` A vector of pressure atmosphere of weather station

`code` The 6 digit code indicates the source of the 6 data columns

`extra` A list of variables need to store

`res` All statistics results store in this slot

`figures` A list to store all plotted figures.

`file.path` The file path for this site.

`data.format` The data format for this site.

`load.later` Whether are records loaded laterly.

`writeWeatherRecords` *Write weather records into file*

Description

Write weather records into file

Write weather records into file

Usage

```
writeWeatherRecords(object, ...)

## S4 method for signature 'WeaAna'
writeWeatherRecords(object, file, cols = NULL)
```

Arguments

`object`	A WeaAna object.
`...`	Not used
`file`	Path of output file.
`cols`	Columns to export. All columns exported if NULL

Value

No return values

`[,WeaAna-method` *Getter to access the weather data at a specific position.*

Description

Getter to access the weather data at a specific position.

Usage

```
## S4 method for signature 'WeaAna'
x[i, j, drop]
```

Arguments

`x`	A WeaAna object.
`i`	the specific position which will access.
`j`	None use parameter.
`drop`	None use parameter.

Value

A WeaAnaSite object at the position i.

Examples

```
library(weaana)
data( "WeatherRecordsDemo" )
records[1]
records[1:2]
records[2:2]
```

Index