Package 'solartime'

November 22, 2024

Title Utilities Dealing with Solar Time Such as Sun Position and Time of Sunrise

Version 0.0.4

Author Thomas Wutzler [aut, cre]

Maintainer Thomas Wutzler <twutz@bgc-jena.mpg.de>

Description Provide utilities to work with solar time,
i.e. where noon is exactly when sun culminates.
Provides functions for computing sun position and times of sunrise and sunset.

Depends R (>= 3.2.0)

URL https://github.com/bgctw/solartime

Imports lubridate

Suggests testthat, knitr, rmarkdown, spelling

License GPL-3

LazyLoad yes

VignetteBuilder knitr

RoxygenNote 7.1.1

Language en-US

NeedsCompilation no

Repository CRAN

Date/Publication 2024-11-22 13:40:01 UTC

Contents

`solartime-package` *solar time utilities.*

Description

Provide utilities to work with solar time, i.e. where noon is exactly when sun culminates. Provides functions for computing sun position and times of sunrise and sunset.

Details

Most fundamental functions are

- corrected fractional hour `getSolarTimeHour` based on `computeSolarToLocalTimeDifference`
- computing position of the sun `computeSunPosition`

On this basis, properties are computed such as

- hour of sunrise and sunset: `computeSunriseHour`,`computeSunsetHour`
- daylength in hours: `computeDayLength`
- flagging times as day or night: `computeIsDayByHour` and `computeIsDayByLocation` and

More utils provide

- get the hours ahead UTC: `getHoursAheadOfUTC`
- get fractional hour of the day: `getFractionalHours`

Also have a look at the package vignettes.

Author(s)

Thomas Wutzler

computeDayLength *computeDayLength*

Description

Compute the Day-length in hours for given time and coordinates

Usage

```
computeDayLength(timestamp, latDeg, ...)
```

Arguments

timestamp	POSIXt vector
latDeg	Latitude in (decimal) degrees
...	further arguments to computeDayLengthDoy

Value

result of computeDayLengthDoy

Author(s)

Thomas Wutzler

computeDayLengthDoy *computeDayLengthDoy*

Description

Compute the Day-length in hours for given time and coordinates

Usage

```
computeDayLengthDoy(doy, latDeg)
```

Arguments

doy	integer vector with day of year [DoY, 1..366], same length as Hour or length 1
latDeg	Latitude in (decimal) degrees

Value

numeric vector of length(doy) giving the time between sunrise and sunset in hours

Author(s)

Thomas Wutzler

Examples

```
doy <- 1:366
plot( computeDayLengthDoy(doy, latDeg = 51) ~ doy)
# north pole: daylength 0 and 24 hours
plot( computeDayLengthDoy( doy, latDeg = +80) ~ doy )
plot( computeDayLengthDoy( doy, latDeg = -80) ~ doy )
```

---

computeIsDayByHour *computeIsDayByHour*

Description

tell for each date, whether its daytime

Usage

```
computeIsDayByHour(date, sunriseHour = 7,
    sunsetHour = 18, duskOffset = 0)
```

Arguments

date	POSIXct vector
sunriseHour	sunrise as fractional hour (0..24) (vector of length date or length 1)
sunsetHour	sunset as fractional hour (vector of length date or length 1)
duskOffset	integer scalar: time in hours after dusk for which records are still regarded as day

Value

logical vector (length(date)): true if its daytime

Author(s)

Thomas Wutzler

computeIsDayByLocation *computeIsDayByLocation*

Description

tell for each timestamp, whether its daytime

Usage

```
computeIsDayByLocation(timestamp, latDeg,
    longDeg, timeZone = getHoursAheadOfUTC(timestamp),
    duskOffset = 0, isCorrectSolartime = TRUE)
```

Arguments

timestamp	POSIXct vector
latDeg	Latitude in (decimal) degrees
longDeg	Longitude in (decimal) degrees
timeZone	Time zone (in hours) ahead of UTC (Central Europe is +1)
duskOffset	integer scalar: time in hours after dusk for which records are still regarded as day
isCorrectSolartime	set to FALSE to omit correction between local time and solar time, e.g. if coordinates cannot be provided

Details

computes hour of sunrise and sunset from given date in timezone hour (assuming dates are given in timezone instead of solartime)

Value

logical vector (length(date)): true if its daytime

Author(s)

Thomas Wutzler

Examples

```
dateSeq <- seq( as.POSIXct("2017-03-20", tz = "Etc/GMT-1")
                ,as.POSIXct("2017-03-21", tz = "Etc/GMT-1")
                , by = "30 min")
tmp <- computeIsDayByLocation(
  dateSeq, latDeg = 50.93, longDeg = 11.59, timeZone = 1)
plot( tmp ~ dateSeq )
```

```
yday <- as.POSIXlt(dateSeq[1])$yday + 1L
sunrise <- computeSunriseHourDoy(
  yday, latDeg = 50.93, longDeg = 11.59, timeZone = 1)
sunset <- computeSunsetHourDoy(
  yday, latDeg = 50.93, longDeg = 11.59, timeZone = 1)
abline( v = trunc(dateSeq[1], units = "days") + c(sunrise,sunset)*3600L )
```

`computeSolarToLocalTimeDifference` *computeSolarToLocalTimeDifference*

Description

computes the time difference in hours between (apparent) solar time and local time

Usage

```
computeSolarToLocalTimeDifference(longDeg,
    timeZone, doy = NA, fracYearInRad = 2 *
        pi * (doy - 1)/365.24)
```

Arguments

`longDeg`	Longitude in (decimal) degrees
`timeZone`	Time zone (in hours) ahead of UTC (Berlin is +1)
`doy`	integer vector with day of year [DoY, 1..366], Specify NA get mean solar time across the year instead of apparent solar time (i.e. with differences throughout the year due to eccentricity of earth orbit)
`fracYearInRad`	may specify instead of doy for efficiency.

Value

time difference in hours to be added to local winter time to get solar time

Author(s)

Thomas Wutzler

Examples

```
# Jena: 50.927222, 11.586111
longDeg <- 11.586
doi <- 1:366
# due to longitude: west of timezone meridian: sun culminates later,
# solar time is less than local time
(localDiff <- computeSolarToLocalTimeDifference(longDeg, 1L)*60)
# taking into account shift during the year due to earth orbit eccentricity
plot( computeSolarToLocalTimeDifference(longDeg, 1L, doi)*60 ~ doi )
abline(h = localDiff)
```

computeSunPosition *computeSunPosition*

Description

Calculate the position of the sun

Usage

```
computeSunPosition(timestamp, latDeg, longDeg)
```

Arguments

timestamp	POSIXct having a valid tzone attribute,
latDeg	Latitude in (decimal) degrees
longDeg	Longitude in (decimal) degrees

Value

as returned by computeSunPositionDoyHour

Author(s)

Thomas Wutzler

---

computeSunPositionDoyHour *computeSunPositionDoyHour*

Description

Compute the position of the sun (solar angle)

Usage

```
computeSunPositionDoyHour(doy, hour, latDeg,
    longDeg = NA, timeZone = NA, isCorrectSolartime = TRUE)
```

Arguments

doy	integer vector with day of year [DoY, 1..366], same length as Hour or length 1
hour	numeric vector with local winter time as decimal hour [0..24)
latDeg	Latitude in (decimal) degrees
longDeg	Longitude in (decimal) degrees
timeZone	Time zone (in hours) ahead of UTC (Central Europe is +1)
isCorrectSolartime	by default corrects hour (given in local winter time) for latitude to solar time (where noon is exactly at 12:00). Set this to FALSE if times are specified already as solar times.

Details

This code assumes that Hour is given in local winter time zone. By default, it corrects by longitude to solar time (where noon is exactly at 12:00). Set argument `isCorrectSolartime` to FALSE to use the given local winter time instead.

Value

named numeric matrix with one row for each time with entries

hour	Solar time in fractional hours after midnight, (or given hour if isCorrectSolartime = FALSE).
declination	Solar declination (rad)
elevation	Solar elevation (rad) with 0 at horizon increasing towards zenith
azimuth	Solar azimuth (rad) with 0 at North increasing eastwards

Author(s)

Thomas Wutzler

Examples

```
computeSunPositionDoyHour(
  160, hour = 0:24, latDeg = 51, longDeg = 13.6, timeZone = 1L)
```

---

computeSunriseHour *computeSunriseHour*

Description

Compute the hour of sunrise for given day and coordinates

Usage

```
computeSunriseHour(timestamp, latDeg, longDeg = NA,
    timeZone = getHoursAheadOfUTC(timestamp),
    ...)
```

Arguments

timestamp	POSIXt vector
latDeg	Latitude in (decimal) degrees
longDeg	Longitude in (decimal) degrees (not required if solar time is sufficient)
timeZone	Time zone (in hours) ahead of UTC (Central Europe is +1) (not required if solar time is sufficient)
...	further arguments to computeSunriseHourDoy

Value

result of computeSunriseHourDoy

Author(s)

Thomas Wutzler

---

computeSunriseHourDoy *computeSunriseHourDoy*

Description

Compute the hour of sunrise for given day and coordinates

Usage

```
computeSunriseHourDoy(doy, latDeg, longDeg = NA,
    timeZone = NA, isCorrectSolartime = TRUE)
```

Arguments

doy	integer vector with day of year [DoY, 1..366]
latDeg	Latitude in (decimal) degrees
longDeg	Longitude in (decimal) degrees (not required if solar time is sufficient)
timeZone	Time zone (in hours) ahead of UTC (Central Europe is +1) (not required if solar time is sufficient)
isCorrectSolartime	sunrise hour is computed first for solar time (where noon is exactly at 12:00) If TRUE (default) then sunrise hour is converted to local winter time, based on timeZone and longitude.

Value

numeric vector of length(doy) giving the time of sunrise in hours after midnight. Polar night is indicated by 12h, polar day by 0h.

Author(s)

Thomas Wutzler

Examples

```
today <-
  as.POSIXlt(Sys.Date())$yday
(sunrise <- computeSunriseHourDoy(today, latDeg = 51, isCorrectSolartime = FALSE))
(sunrise <- computeSunriseHourDoy(today, latDeg = 51, longDeg = 11.586, timeZone = +1))
# elevation near zero
computeSunPositionDoyHour(160, sunrise, latDeg = 51, isCorrectSolartime = FALSE)
#
doy <- 1:366
plot( computeSunriseHourDoy(doy, latDeg = 51, isCorrectSolartime = FALSE) ~ doy )
# north pole: daylength 0 and 24 hours
plot( computeSunriseHourDoy( doy, latDeg = +80, isCorrectSolartime = FALSE) ~ doy )
plot( computeSunriseHourDoy( doy, latDeg = -80, isCorrectSolartime = FALSE) ~ doy )
```

`computeSunsetHour` *computeSunsetHour*

Description

Compute the hour of sunrise for given day and coordinates

Usage

```
computeSunsetHour(timestamp, latDeg, longDeg = NA,
    timeZone = getHoursAheadOfUTC(timestamp),
    ...)
```

Arguments

`timestamp`	POSIXt vector
`latDeg`	Latitude in (decimal) degrees
`longDeg`	Longitude in (decimal) degrees (not required if solar time is sufficient)
`timeZone`	Time zone (in hours) ahead of UTC (Central Europe is +1) (not required if solar time is sufficient)
`...`	further arguments to `computeSunsetHourDoy`

Value

result of `computeSunsetHourDoy`

Author(s)

Thomas Wutzler

`computeSunsetHourDoy` *computeSunsetHourDoy*

Description

Compute the hour of sunrise for given day and coordinates

Usage

```
computeSunsetHourDoy(doy, latDeg, longDeg = NA,
    timeZone = NA, isCorrectSolartime = TRUE)
```

Arguments

`doy`	integer vector with day of year [DoY, 1..366]
`latDeg`	Latitude in (decimal) degrees
`longDeg`	Longitude in (decimal) degrees (not required if solar time is sufficient)
`timeZone`	Time zone (in hours) ahead of UTC (Central Europe is +1) (not required if solar time is sufficient)
`isCorrectSolartime`	sunrise hour is computed first for solar time (where noon is exactly at 12:00) If TRUE (default) then sunrise hour is converted to local winter time, based on timeZone and longitude.

Value

numeric vector of length(doy) giving the time of sunset in hours after midnight. Polar night is indicated by 12h, polar day by 24h.

Author(s)

Thomas Wutzler

Examples

```
today <-
  as.POSIXlt(Sys.Date())$yday
(sunset <- computeSunsetHourDoy(today, latDeg = 51, isCorrectSolartime = FALSE))
(sunset <- computeSunsetHourDoy(today, latDeg = 51, longDeg = 11.586, timeZone = +1))
#
doy <- 1:366
plot( computeSunsetHourDoy(doy, latDeg = 51, isCorrectSolartime = FALSE) ~ doy )
# north pole: daylength 0 and 24 hours
plot( computeSunsetHourDoy( doy, latDeg = +80, isCorrectSolartime = FALSE) ~ doy )
plot( computeSunsetHourDoy( doy, latDeg = -80, isCorrectSolartime = FALSE) ~ doy )
```

getFractionalHours *getFractionalHours*

Description

get the time difference to previous midnight in fractional hours

Usage

```
getFractionalHours(timestamp)
```

Arguments

timestamp	POSIXt vector

Value

numeric vector of fractional hours

Author(s)

Thomas Wutzler

getHoursAheadOfUTC *getHoursAheadOfUTC*

Description

get the time difference to UTC in hours

Usage

```
getHoursAheadOfUTC(timestamp)
```

Arguments

timestamp	POSIXt vector

Value

integer vector of how many hours noon of timestamp is ahead of noon in UTC

Author(s)

Thomas Wutzler

`getSolarTimeHour` *getSolarTimeHour*

Description

Get the fractional hour of solar time

Usage

```
getSolarTimeHour(timestamp, longDeg)
```

Arguments

`timestamp`	POSIXt vector in local time
`longDeg`	Longitude in (decimal) degrees

Value

fractional hour corrected by difference to local time

Author(s)

Thomas Wutzler

`setLocalTimeZone` *setLocalTimeZone*

Description

modify tzone attribute of timestamp to 'GMT+x' for local to given longitude

Usage

```
setLocalTimeZone(timestamp, longDeg)
```

Arguments

`timestamp`	POSIXct
`longDeg`	Longitude in (decimal) degrees

Value

`timestamp` with modified tzone attribute. Its the same time point expressed in another time zone. E.g. "2019-04-04 00:00:00 UTC" becomes "2019-04-04 10:00:00 +10" for a longitude of +150 (Sydney, Australia)

Author(s)

Thomas Wutzler

Index