Package 'HeatStressR'

September 18, 2026

Type Package

Title Calculate Heat Stress Indices

Version 2.4.0

Date 2026-09-02

Description Calculates heat-stress indices from meteorological observations, including the physically based wet-bulb globe temperature model described by Liljegren et al. (2008) <doi:10.1080/15459620802310770>. The package provides an independently maintained R implementation with row-level diagnostics, configurable physical controls, and batch processing for the Liljegren method; it is not a bitwise-compatible port of the original program, and cross-implementation differences are expected.

License GPL-3

URL https://github.com/zyf0717/HeatStressR

BugReports https://github.com/zyf0717/HeatStressR/issues

RoxygenNote 8.0.0

Depends R (>= 3.4.0)

Suggests pkgload, testthat (>= 3.0.0)

Config/testthat/edition 3

Imports stats, assertthat, parallel

Encoding UTF-8

NeedsCompilation no

Author Yifei Zheng [aut, cre] (Maintainer of this fork), Ana Casanueva [aut] (Original package author)

Maintainer Yifei Zheng <zyf0717@gmail.com>

Repository CRAN

Date/Publication 2026-09-18 07:00:21 UTC

Contents

`"data_obs"` *Meteorological input variables for the heat stress indices.*

Description

A dataset containing the input variables for the heat stress indices as examples. It contains observed daily data from the ECA&D dataset for summer 2003 and one station (Salamanca, Spain).

Usage

```
data(data_obs)
```

Format

A data frame with 92 rows (days) and 6 variables. The variables are as follows:

- Dates: Dates vector of dates for summer 2003.
- tasmean: numeric vector of daily mean temperature.
- dewp: numeric vector od daily mean dew point temperature.
- hurs: numeric vector of daily mean relative humidity.
- wind: numeric vector of daily mean wind speed.
- solar: numeric vector of daily mean solar radiation. These values come from the closest g grid box from the SARAH satellite dataset (DOI:10.5676/EUM_SAF_CM/SARAH/V001).

Source

https://www.ecad.eu/index.php and DOI:10.5676/EUM_SAF_CM/SARAH/V001

`"data_wbgt.Bernard"` *Example for Wet Bulb Globe Temperature in the shade (Bernard et al. 1999).*

Description

Dataset to check the result of `wbgt.Bernard` with the testing data `obs_data` for summer 2003 and one station (Salamanca, Spain).

Usage

```
data(data_wbgt.Bernard)
```

Format

List of two elements:

- data: numeric vector with the WBGT values in degC.
- Tpwb: numeric vector with the phychrometric wet bulb temperature (Tpwb) in degC.

"data_wbgt.Liljegren" *Example for Wet Bulb Globe Temperature in the sun (Liljegren et al. 2008).*

Description

Dataset to check the result of `wbgt.Liljegren` with the testing data `obs_data` for summer 2003 and one station (Salamanca, Spain).

Usage

```
data(data_wbgt.Liljegren)
```

Format

List of three elements:

- data: numeric vector with the WBGT values in degC.
- Tpwb: natural wet bulb temperature (Tnwb) in degC.
- Tg: globe temperature in degC.

"data_wbt.Stull" *Example for Wet Bulb Temperature (Stull 2011).*

Description

Dataset to check the result of `wbt.Stull` with the testing data `obs_data` for summer 2003 and one station (Salamanca, Spain).

Usage

```
data(data_wbt.Stull)
```

Format

Numeric vector with 92 values (days).

apparentTemp *Calculation of the apparent temperature.*

Description

Calculation of the apparent temperature from temperature, relative humidity and wind.

Usage

```
apparentTemp(tas, hurs, wind)
```

Arguments

tas	vector of air temperature in degC.
hurs	vector of relative humidity in %.
wind	vector of wind at 10m in m/s.

Details

Formula based on air temperature, relative humidity and wind only, as it is calculated in Steadman 1994, Buzan et al. 2015 GMD and references therein. There is a version including radiation (net radiation absorved per unit area of human body surface), but it is not implemented here.

Value

Apparent temperature in degC.

Author(s)

A.Casanueva (22.03.2018).

Examples

```
# load the meteorological variables for example data in Salamanca:
data("data_obs")
at <- apparentTemp(data_obs$tasmean, hurs=data_obs$hurs, wind= data_obs$wind)
```

`calZenith` *Calculate zenith angle in degrees.*

Description

Calculate zenith angle in degrees.

Usage

```
calZenith(dates, lon, lat, hour = FALSE, solar_time = "timestamp")
```

Arguments

`dates`	vector of dates, `POSIXct`/`POSIXlt` instants, or ISO 8601 datetime strings. Use timezone-aware `POSIXct` for high-throughput calls. With `solar_time = "timestamp"`, offset-bearing ISO 8601 strings are interpreted as instants and normalized to UTC; strings are parsed on every call. `solar_time = "date_noon"` evaluates each date at 12:00 UTC.
`lon`	single numeric longitude for the location, in degrees.
`lat`	single numeric latitude for the location, in degrees.
`hour`	legacy logical solar-time selector. Use `solar_time` in new code.
`solar_time`	`"timestamp"` (the default) uses each full timestamp; `"date_noon"` evaluates each date at 12:00 UTC. The legacy `hour` argument remains supported when `solar_time` is omitted.

Details

`lon` and `lat` must be finite scalar values within their standard ranges. Solar time incorporates longitude and the equation of time. Missing dates return `NA` in the corresponding output position.

Value

Numeric vector of zenith angles in degrees, aligned with `dates`.

Author(s)

Anke Duguay-Tetzlaff, Translated to R by Ana Casanueva (17.01.2017)

Examples

```
calZenith("1981-06-15", -5.66, 40.96)
calZenith("1981-06-15 10:00:00", -5.66, 40.96, solar_time = "timestamp")
calZenith("1981-06-15T18:00:00+08:00", -5.66, 40.96, solar_time = "timestamp")
```

`degToRad` *Convert degree angle to radians.*

Description

Convert degree angle to radians.

Usage

```
degToRad(angleDeg)
```

Arguments

`angleDeg`	angle in degree.

Value

Angle in radians.

Author(s)

Ana Casanueva (10.01.2017).

`dewp2hurs` *Calculation of relative humidity from temperature and dewpoint temperature.*

Description

Calculation of relative humidity from temperature and dewpoint temperature.

Usage

```
dewp2hurs(tas, dewp)
```

Arguments

`tas`	vector of temperature in degC
`dewp`	vector of dewpoint temperature in degC

Details

Formulation from Dosseger et al. 1992. Formula 99 in MCH document.

Value

relative humidity in

Author(s)

Ana Casanueva (11.08.2016)

`diffusivity` *Compute the diffusivity of water vapor in air.*

Description

Compute the diffusivity of water vapor in air.

Usage

```
diffusivity(Tk, Pair)
```

Arguments

`Tk`	value of air temperature in Kelvin.
`Pair`	value of air pressure in hPa.

Details

Reference: BSL, page 505.

Value

Diffusivity of water vapor in air, m2/s.

Author(s)

Ana Casanueva (05.01.2017).

`discomInd` *Calculation of the discomfort index.*

Description

Calculation of the discomfort index from temperature and relative humidity.

Usage

```
discomInd(tas, hurs)
```

Arguments

`tas`	vector of air temperature in degC.
`hurs`	vector of relative humidity in %.

Details

Formula based on air temperature and relative humidity, as it is calculated in Coccolo et al. 2016 and references therein.

Value

Discomfort index in degC.

Author(s)

A.Casanueva (22.03.2018).

Examples

```
# load the meteorological variables for example data in Salamanca:
data("data_obs")
di <- discomInd(data_obs$tasmean, hurs=data_obs$hurs)
```

`effectiveTemp` *Calculation of the effective temperature.*

Description

Calculation of the effective temperature from temperature, relative humidity and wind.

Usage

```
effectiveTemp(tas, hurs, wind)
```

Arguments

`tas`	vector of air temperature in degC.
`hurs`	vector of relative humidity in %.
`wind`	vector of wind at 10m in m/s.

Details

Formula based on air temperature, relative humidity and wind, as it is calculated in Coccolo et al. 2016 and references therein.

Value

Effective temperature in degC.

Author(s)

A.Casanueva (22.03.2018).

Examples

```
# load the meteorological variables for example data in Salamanca:
data("data_obs")
et <- effectiveTemp(data_obs$tasmean, hurs=data_obs$hurs, wind=data_obs$wind)
```

`emis_atm` *Calculate the atmospheric emissivity.*

Description

Calculate the atmospheric emissivity.

Usage

```
emis_atm(Tk, RH)
```

Arguments

`Tk`	value of air temperature in Kelvin.
`RH`	value of relative humidity in fraction.

Details

Reference: Oke (2nd edition), page 373.

Value

atmospheric emissivity.

Author(s)

Ana Casanueva (05.01.2017).

`esat` *Calculate the saturation vapor pressure (hPa) over water.*

Description

Calculate the saturation vapor pressure (hPa) over water.

Usage

```
esat(Tk)
```

Arguments

Tk	value of air temperature in Kelvin.

Details

Reference: Buck's (1981) approximation (eqn 3) of Wexler's (1976) formulae over liquid water.

Value

saturation vapor pressure (hPa).

Author(s)

Ana Casanueva (05.01.2017).

---

fTg *Calculation of the globe temperature.*

Description

Calculation of the globe temperature.

Usage

```
fTg(
  tas,
  relh,
  Pair,
  wind,
  min.speed,
  radiation,
  propDirect,
  zenith,
  SurfAlbedo = 0.45,
  tolerance = 1e-04,
  globe_diameter = 0.0508
)
```

Arguments

tas	vector of temperature in degC.
relh	vector of relative humidity in %.
Pair	value of air pressure in hPa.
wind	vector of wind speed in m/s.
min.speed	value of minimum wind speed in m/s.
radiation	vector of solar shortwave downwelling radiation in W/m2.

propDirect	proportion of direct radiation = direct/(diffuse + direct).
zenith	zenith angle in radians.
SurfAlbedo	(optional) surface albedo. Default: 0.45.
tolerance	(optional) tolerance value for the iteration. Default: 1e-4.
globe_diameter	black-globe diameter in m. Default: 0.0508.

Details

Original fortran code by James C. Liljegren, translated by Bruno Lemke into Visual Basic (VBA). Uses an adaptively bracketed signed heat-balance residual.

Value

Globe temperature in degC.

Author(s)

Ana Casanueva (05.01.2017).

---

fTnwb *Calculation of the natural wet bulb temperature.*

Description

Calculation of the natural wet bulb temperature.

Usage

```
fTnwb(
  tas,
  dewp,
  relh,
  Pair,
  wind,
  min.speed,
  radiation,
  propDirect,
  zenith,
  irad = 1,
  SurfAlbedo = 0.45,
  tolerance = 1e-04
)
```

Arguments

tas	vector of temperature in degC.
dewp	vector of dewpoint temperature in degC.
relh	vector of relative humidity in %.
Pair	value of air pressure in hPa.
wind	vector of wind speed in m/s.
min.speed	value of minimum wind speed in m/s.
radiation	vector of solar shortwave downwelling radiation in W/m2.
propDirect	proportion of direct radiation = direct/(diffuse + direct).
zenith	zenith angle in radians.
irad	(optional): include radiation (1) or not (irad=0, psychrometric web bulb temp). Default: 1.
SurfAlbedo	(optional) surface albedo. Default: 0.45.
tolerance	(optional) tolerance value for the iteration. Default: 1e-4.

Details

Original fortran code by James C. Liljegren, translated by Bruno Lemke into Visual Basic (VBA).

Value

Natural wet bulb globe temperature in degC.

Author(s)

Ana Casanueva (05.01.2017).

HeatStressR *HeatStressR*

Description

Calculate heat stress indices

Details

The package HeatStressR calculates heat-stress indices from meteorological observations and exposes both index-level functions and lower-level physical components.

The following calculation methods are implemented:

- wbt.Stull: Calculation of the Wet Bulb Temperature (Stull 2011).
- wbgt.Bernard: Calculation of the Wet Bulb Globe Temperature in the shade (Bernard et al. 1999)

- `wbgt.Liljegren`: Calculation of the Wet Bulb Globe Temperature in the sun (Liljegren et al. 2008)
- `swbgt`: Calculation of the simplified wet bulb globe temperature (Buzan et al. 2015 and references therein).
- `apparentTemp`: Calculation of the apparent temperature (Buzan et al. 2015 and references therein).
- `effectiveTemp`: Calculation of the effective temperature (Coccolo et al. 2016 and references therein).
- `humidex`: Calculation of the humidex (Buzan et al. 2015 and references therein).
- `discomInd`: Calculation of the discomfort index (Coccolo et al. 2016 and references therein).
- `hi`: Calculation of the heat index (NOAA, Rothfusz 1990).
- `heat_indices`: Fused calculation of multiple non-Liljegren indices from shared observations.

`wbgt.Liljegren()` implements the outdoor Liljegren wet-bulb globe temperature model in R. Its vectorized batch engine is the default and can use explicitly requested base R PSOCK workers; the scalar engine remains available as a reference implementation. Pressure and documented physical constants are configurable. Set `diagnostics = TRUE` to obtain row-aligned input and solver metadata. This is not a bitwise-compatible port of the original Liljegren program, and cross-implementation differences are expected.

Invalid inputs and numerical solver failures are distinct. Complete WBGT is `NA` unless both globe and natural-wet-bulb temperatures validate, while an independently valid component can be retained. Diagnostic vectors remain aligned with the supplied meteorological rows.

HeatStressR is an independently maintained fork of the HeatStress package. Ana Casanueva made the original R translation; this fork changes the R package implementation by adding explicit numerical controls, row-level diagnostics, and optional batch execution. These changes are not a claim that HeatStressR improves on or supersedes the original Liljegren implementation. It is maintained by Yifei Zheng and is not affiliated with the original project or its authors. The source repository is https://github.com/zyf0717/HeatStressR.

CRAN checks package portability and software quality; users remain responsible for matching the methodological assumptions of a chosen index to their application. To cite the package and the Liljegren model, use `citation("HeatStressR")`.

Check the details of the indices and input variables with `indexShow()`. Use `heat_indices()` to calculate multiple closed-form indices from the same observations while reusing shared intermediate calculations.

`heat_indices` *Calculate multiple heat indices from shared observations.*

Description

Calculates selected non-Liljegren heat indices while reusing validation and vapour pressure calculations across the requested indices.

Usage

```
heat_indices(tas, hurs, wind = NULL, dewp = NULL, indices = NULL)
```

Arguments

tas	vector of air temperature in degC.
hurs	vector of relative humidity in %.
wind	optional vector of wind at 10m in m/s. Required by apparentTemp and effectiveTemp.
dewp	optional vector of dew point temperature in degC. Required only when wbgt.Bernard is requested.
indices	optional character vector of requested indices. By default, all indices supported by the supplied inputs are selected. wbgt.Bernard is never selected automatically.

Details

The closed-form indices are calculated directly from shared inputs. Without wind, the default excludes apparentTemp and effectiveTemp; supplying wind selects the complete default set. Vapour pressure is calculated once when one or more of swbgt, apparentTemp, and humidex is requested.

Value

A data frame with one row per input observation and one column per requested index. The optional wbgt.Bernard column contains the WBGT value; call wbgt.Bernard() directly to obtain Tpwb too.

Examples

```
heat_indices(
  tas = c(25, 30), hurs = c(60, 70), wind = c(1, 2),
  indices = c("humidex", "hi", "apparentTemp")
)
```

---

hi *Calculation of the heat index.*

Description

Calculation of the heat index from temperature and relative humidity.

Usage

```
hi(tas, hurs)
```

Arguments

tas	vector of air temperature in degC.
hurs	vector of relative humidity in %.

Details

Formula based on air temperature and relative humidity, following Rothfusz 1990 (National Weather Service Technical Attachment, SR 90-23). The NWS equations and adjustments are evaluated in degrees Fahrenheit internally and the final heat index is returned in degrees Celsius. This implementation includes some adjustments for high and low relative humidity values. Also, the original formula is not appropriate for low temperatures and heat index values. In those cases, a simpler formula is applied to calculate values consistent with Steadman's results. See: https://www.wpc.ncep.noaa.gov/html/heatindex_equation and https://github.com/ecmwf/thermofeel/blob/master/thermofeel/thermofeel.py#L782

Value

Heat index in degC.

Author(s)

A.Casanueva (22.03.2018). Modified in 12.08.2025.

Examples

```
# load the meteorological variables for example data in Salamanca:
data("data_obs")
heatindex <- hi(data_obs$tasmean, hurs=data_obs$hurs)
```

---

humidex *Calculation of humidex.*

Description

Calculation of humidex from temperature and relative humidity.

Usage

```
humidex(tas, hurs)
```

Arguments

tas	vector of air temperature in degC.
hurs	vector of relative humidity in %.

Details

Formula based on air temperature and relative humidity, as it is calculated in Buzan 2015 GMD and references therein.

Value

Humidex values in degC.

Author(s)

A.Casanueva (22.03.2018).

Examples

```
# load the meteorological variables for example data in Salamanca:
data("data_obs")
hum <- humidex(data_obs$tasmean, hurs=data_obs$hurs)
```

---

h_cylinder_in_air *Calculate the convective heat transfer coefficient for a long cylinder in cross flow.*

Description

Calculate the convective heat transfer coefficient for a long cylinder in cross flow.

Usage

```
h_cylinder_in_air(Tk, Pair, speed, min.speed, diam.wick)
```

Arguments

Tk	value of air temperature in Kelvin.
Pair	value of air pressure in hPa.
speed	value of wind speed in m/s.
min.speed	value of minimum wind speed in m/s.
diam.wick	diameter of the cylinder in m.

Details

Reference: Bedingfield and Drew, eqn 32.

Value

Convective heat transfer coefficient for a long cylinder, W/(m2 K).

Author(s)

Ana Casanueva (05.01.2017).

`h_evap` *Calculate the heat of evaporation, J/(kg K).*

Description

Calculate the heat of evaporation, J/(kg K), for temperature in the range 283-313 K.

Usage

```
h_evap(Tk)
```

Arguments

`Tk`	value of air temperature in Kelvin.

Details

Reference: Van Wylen and Sonntag, Table A.1.1

Value

Heat of evaporation, J/(kg K).

Author(s)

Ana Casanueva (05.01.2017).

`h_sphere_in_air` *Calculate the convective heat transfer coefficient for flow around a sphere.*

Description

Calculate the convective heat transfer coefficient for flow around a sphere.

Usage

```
h_sphere_in_air(Tk, Pair, speed, min.speed, diam.globe)
```

Arguments

Tk	value of air temperature in Kelvin.
Pair	value of air pressure in hPa.
speed	value of wind speed in m/s.
min.speed	value of minimum wind speed in m/s.
diam.globe	diameter of the sphere in m.

Details

Reference: Bird, Stewart, and Lightfoot (BSL), page 409.

Value

Convective heat transfer coefficient for flow around a sphere, W/(m2 K).

Author(s)

Ana Casanueva (05.01.2017).

---

indexShow *List all available heat indices*

Description

Print a table with a summary of the available single-index calculations. The fused `heat_indices()` helper is documented separately because it returns multiple requested indices.

Usage

```
indexShow()
```

Value

Print a table on the screen with the following columns:

- **code**: Code of the index.
- **longname**: Long description of the index
- **index.fun**: The name of the internal function used to calculate it
- **tas, dewp, hurs, wind, radiation**: A logical value (0/1) indicating the input variables required for index calculation. Temperature and either dew point temperature or relative humidity need to be always provided.
- **units**: The units of the index.

Author(s)

A. Casanueva

`is.leapyear` *Check whether a year is a leap year.*

Description

Check whether a year is a leap year.

Usage

```
is.leapyear(year)
```

Arguments

`year`	to be checked.

Value

logical, TRUE/FALSE.

Author(s)

Sven Kotlarski (20.12.2016).

`radToDeg` *Convert radian angle to degrees.*

Description

Convert radian angle to degrees.

Usage

```
radToDeg(angleRad)
```

Arguments

`angleRad`	angle in radians.

Value

Angle in degrees.

Author(s)

Ana Casanueva (10.01.2017).

swbgt *Calculation of the simplified wet bulb globe temperature.*

Description

Calculation of the simplified wet bulb globe temperature from temperature and relative humidity.

Usage

```
swbgt(tas, hurs)
```

Arguments

tas	vector of air temperature in degC.
hurs	vector of relative humidity in %.

Details

Formula based on air temperature and relative humidity, as it is calculated in Buzan et al. 2015 GMD with a small correction in the constants from Lemke and Kjellstrom 2012.

Value

Simplified wet bulb globe temperature in degC.

Author(s)

A.Casanueva (22.03.2018).

Examples

```
# load the meteorological variables for example data in Salamanca:
data("data_obs")
swbgt <- swbgt(data_obs$tasmean, hurs=data_obs$hurs)
```

tashurs2vap.pres *Calculation of vapour pressure.*

Description

Calculation of vapour pressure from temperature and relative humidity

Usage

```
tashurs2vap.pres(tas, hurs)
```

Arguments

tas	vector of air temperature in degC.
hurs	vector of relative humidity in %.

Details

Formulation from Dosseger et al. 1992. Formula 16 in MCH document. Relative humidity values above 100% are clamped to 100% for backwards compatibility. Higher-level index functions reject those inputs instead.

Value

Vapour pressure in hPa.

Author(s)

A.Casanueva (11.08.2016).

Examples

```
# load the meteorological variables for example data in Salamanca:
data("data_obs")
vp <- tashurs2vap.pres(data_obs$tasmean, hurs=data_obs$hurs)
```

---

thermal_cond *Calculate the thermal conductivity of air, W/(m K).*

Description

Calculate the thermal conductivity of air, W/(m K).

Usage

```
thermal_cond(Tk)
```

Arguments

Tk	value of air temperature in Kelvin.

Details

Reference: BSL, page 257.

Value

Thermal conductivity of air, W/(m K).

Author(s)

Ana Casanueva (05.01.2017).

`viscosity` *Compute the viscosity of air, kg/(m s).*

Description

Compute the viscosity of air, kg/(m s) given temperature (K).

Usage

```
viscosity(Tk)
```

Arguments

`Tk`	value of air temperature in Kelvin.

Details

Reference: BSL, page 23.

Value

viscosity of air, kg/(m s).

Author(s)

Ana Casanueva (05.01.2017).

`wbgt.Bernard` *Calculation of wet bulb globe temperature, following Bernard's method.*

Description

Calculation of wet bulb globe temperature from air temperature and dew point temperature. This corresponds to the implementation for indoors or shadow conditions.

Usage

```
wbgt.Bernard(tas, dewp, tolerance = 1e-04, noNAs = TRUE, swap = FALSE)
```

Arguments

tas	vector of air temperature in degC.
dewp	vector of dew point temperature in degC.
tolerance	(optional): maximum final bracket width for the psychrometric wet-bulb solution. Default: 1e-4.
noNAs	logical, should NAs be introduced when dewp>tas? If TRUE specify how to deal in those cases (swap argument)
swap	logical, should tas >= dewp be enforced by swapping? Otherwise, dewp is set to tas. This argument is needed when noNAs=T.

Details

Based on Lemke and Kjellstrom 2012, using the formulation from Bernard et al. 1999. The psychrometric wet-bulb temperature is solved with vectorized bisection on the physical interval from dew point to air temperature.

Value

A list of:

$data: wet bulb globe temperature in degC.

$Tpwb: phychrometric wet bulb temperature (Tpwb) in degC.

Author(s)

A.Casanueva, P. Noti, J. Bhend (21.02.2017).

Examples

```
# load the meteorological variables for example data in Salamanca:
data("data_obs")
wbgt.indoors <- wbgt.Bernard(tas=data_obs$tasmean, dewp=data_obs$dewp)
```

---

wbgt.Liljegren *Calculation of wet bulb globe temperature, following Liljegren's method.*

Description

Calculation of wet bulb globe temperature from air temperature, dew point temperature, radiation and wind.

Usage

```
wbgt.Liljegren(
  tas,
  dewp,
  wind,
  radiation,
  dates,
  lon,
  lat,
  tolerance = 1e-04,
  noNAs = TRUE,
  swap = FALSE,
  hour = FALSE,
  engine = c("batch", "scalar"),
  diagnostics = FALSE,
  root_tolerance = NULL,
  residual_tolerance = NULL,
  dewpoint_tolerance = NULL,
  pressure = 1010,
  surface_albedo = 0.45,
  globe_diameter = 0.0508,
  min_wind_speed = 0.13,
  workers = 1L,
  solar_time = "timestamp",
  direct_fraction = 0.8
)
```

Arguments

tas	vector of temperature in degC.
dewp	vector of dewpoint temperature in degC.
wind	vector of wind speed at 2 m above ground in m/s. Wind-height adjustment is not performed internally.
radiation	vector of solar shortwave downwelling radiation in W/m2.
dates	vector of dates, POSIXct/POSIXlt instants, or ISO 8601 datetime strings. Use timezone-aware POSIXct for high-throughput calls. With solar_time = "timestamp", offset-bearing ISO 8601 strings are normalized to UTC; strings are parsed on every call. Values must have the same length and row order as the meteorological input vectors.
lon	numeric longitude in degrees. Supply one value for a fixed location or a vector aligned with the meteorological inputs.
lat	numeric latitude in degrees. Supply one value for a fixed location or a vector aligned with the meteorological inputs.
tolerance	Legacy tolerance control. When the independent controls are not supplied, it maps to root precision tolerance * 0.01, residual acceptance tolerance, and dewpoint validation tolerance.

noNAs logical, should NAs be introduced when dewp>tas? If TRUE specify how to deal in those cases (swap argument)

swap logical, should tas >= dewp be enforced by swapping? Otherwise, dewp is set to tas. This argument is needed when noNAs=T.

hour legacy logical solar-time selector. Use solar_time in new code.

engine Numerical solver engine. "batch" is the default vectorized safeguarded root solver with automatic scalar fallback for unresolved rows. "scalar" selects the reference R implementation.

diagnostics logical; return solver and preprocessing metadata in addition to the usual result.

root_tolerance numerical precision (K) used to locate heat-balance roots.

residual_tolerance
maximum accepted absolute heat-balance residual (K). Must be greater than zero and no greater than 0.01.

dewpoint_tolerance
permitted dewpoint-versus-air-temperature difference (degrees C) used by the dewpoint policy.

pressure atmospheric pressure in hPa. Supply one value or a vector aligned with the meteorological inputs; defaults to 1010 hPa.

surface_albedo surface shortwave albedo. Defaults to 0.45, matching the original Liljegren C implementation.

globe_diameter black-globe diameter in m. Defaults to 0.0508 m, matching the original Liljegren C implementation.

min_wind_speed lower bound applied to wind speed in m/s. Defaults to 0.13 m/s, matching the original Liljegren C implementation.

workers number of PSOCK worker processes for engine = "batch". Must be an integer from 1 through the currently permitted logical CPU count. The default of 1 preserves sequential batch execution; values greater than 1 use up to the requested number of workers, capped at the number of input rows. In an R check environment with _R_CHECK_LIMIT_CORES_ = "true", no more than two workers are permitted. Workers calculate solar geometry, preprocess their own coordinate-aware meteorological chunk, including humidity, then solve and assemble local WBGT results.

solar_time "timestamp" (the default) uses each full timestamp; "date_noon" evaluates each date at 12:00 UTC. The legacy hour argument remains supported when solar_time is omitted.

direct_fraction
proportion of supplied shortwave radiation treated as direct, ‘direct / (direct + diffuse)‘. Supply one value or a vector aligned with the meteorological inputs; defaults to 0.8.

Details

This corresponds to the implementation for outdoors or in the sun conditions described by Liljegren et al. (2008), doi:10.1080/15459620802310770. The original Fortran code was written by James C. Liljegren, translated to Visual Basic (VBA) by Bruno Lemke, and translated to R by Ana Casanueva.

HeatStressR is an independently maintained fork and is not affiliated with the original project or its authors.

The batch engine is the default implementation. It uses explicitly requested base R PSOCK workers when `workers > 1`; no workers are created by default. The scalar engine remains available as a reference implementation. Pressure, surface albedo, globe diameter, minimum wind speed, and direct-radiation fraction are configurable. Solar positions use the supplied timestamp, latitude, longitude, and the equation of time. Radiation is zeroed when the computed solar elevation is not positive. Solar geometry deduplicates timestamp-only terms for repeated instants, then applies one vectorized row-aligned longitude/latitude projection. The function evaluates aligned instantaneous meteorological states; interval alignment, timestamp conversion, wind-height adjustment, and radiation quality control remain caller responsibilities. When direct and diffuse radiation are available, supply their direct share with `direct_fraction`.

`dates` must have the same length and row order as the meteorological input vectors. Root-location precision, residual validation, and dewpoint validation are controlled independently. Relaxing `residual_tolerance` accepts only candidate roots that were found; it cannot recover unbracketed or non-finite solves. Complete WBGT requires both validated component roots, but a validated Tg or Tnwb value is retained when the other component fails. Solar forcing is set to zero when the solar elevation is not positive. Diagnostics flag supplied radiation greater than 15 W/m2 at zenith angles greater than 1.54 radians as `solar_geometry_mismatch`. Each component diagnostic includes convergence, evaluations, final residual, failure reason, fallback use, initial/final brackets, endpoint residuals, and resolved root/residual tolerances. `complete_wbgt` identifies rows with both validated component roots. With `diagnostics = TRUE`, all row-level diagnostic vectors match the input length. `input_status` describes filtering, while per-solver `converged` and `fallback_reason` describe numerical solving. Preprocessing is recorded by `wind_clamped`, `radiation_clamped`, `radiation_zeroed_below_horizon`, and `dewpoint_adjusted`. `workers` reports the effective worker count and `requested_workers` reports the supplied count.

Agreement with another implementation requires matching pressure, wind-height treatment, timestamp convention, solar-position method, radiation partitioning, and failure semantics. This function is not a bitwise-compatible port of the original C implementation, and differences from other implementations are expected. These differences are not intended as a claim that this R implementation improves on or supersedes the original Liljegren program.

Value

A list of:

$data: wet bulb globe temperature in degC

$Tnwb: natural wet bulb temperature (Tnwb) in degC

$Tg: globe temperature in degC

Author(s)

Original R translation: Ana Casanueva (2017). Current fork maintenance and modifications: Yifei Zheng.

Examples

```
times <- as.POSIXct(
```

```
  c("2024-06-01 12:00:00", "2024-06-01 13:00:00"),
  tz = "UTC"
)
result <- wbgt.Liljegren(
  tas = c(30, 31), dewp = c(22, 22.5), wind = c(1.5, 2),
  radiation = c(700, 750), dates = times, lon = 0, lat = 15,
  direct_fraction = c(0.6, 0.8),
  solar_time = "timestamp"
)
result$data


result_parallel <- wbgt.Liljegren(
  tas = c(30, 31), dewp = c(22, 22.5), wind = c(1.5, 2),
  radiation = c(700, 750), dates = times, lon = 0, lat = 15,
  direct_fraction = c(0.6, 0.8),
  solar_time = "timestamp",
  engine = "batch", workers = 2
)
result_parallel$data
```

`wbt.Stull` *Calculation of wet bulb temperature, following Stull's method.*

Description

Calculation of wet bulb temperature from temperature and relative humidity.

Usage

```
wbt.Stull(tas, hurs)
```

Arguments

`tas`	vector of air temperature in degC.
`hurs`	vector of relative humidity in %.

Details

Formulation from Stull 2011, Journal of Applied Meteorology and Climatology.

Value

Wet bulb temperature in degC.

Author(s)

A.Casanueva (15.08.2016).

Examples

```
# load the meteorological variables for example data in Salamanca:
data("data_obs")
wbt <- wbt.Stull(data_obs$tasmean, hurs=data_obs$hurs)
```

Index