Package ‘weathertools’

January 12, 2026

Title Weather and Atmospheric Calculation Utilities

Version 0.1.0

Description Utility functions for computing common weather and
atmospheric quantities such as humidity, vapor pressure, and
temperature-derived metrics.

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Imports Rcpp,
RcppParallel,
data.table
zoo

LinkingTo Rcpp,
RcppParallel

Contents

`avgwdir` *Moving-window average wind direction*

Description

Computes a rolling-mean wind direction using vector (u/v-style) averaging: directions are converted to north/south and east/west components, summed in a rolling window with weights given by wind speed, and converted back to a direction in degrees.

Usage

```
avgwdir(winddirectionsDeg, windspeeds, movingWindow = 10, na.pad = TRUE)
```

Arguments

`winddirectionsDeg`	Numeric vector. Wind direction in degrees (0–360).
`windspeeds`	Numeric vector. Wind speed (same length as `winddirectionsDeg`).
`movingWindow`	Integer. Window size (number of samples) for the rolling sum.
`na.pad`	Logical. If `TRUE`, pads the leading window with `NA` values to preserve input length; if `FALSE`, returns only the fully-defined values.

Details

This function performs a rolling sum on weighted wind components and converts back to a direction via `atan2()`. When `na.pad = TRUE`, the result length equals the input length.

Value

Numeric vector of averaged wind direction degrees in [0, 360).

Examples

```
wd  <- c(350, 10, 15, 20, 25)
wsp <- c(5,   5,  5,  5,  5)
avgwdir(wd, wsp, movingWindow = 3)
```

`calcHI` *Compute Heat Index (fast parallel C++ core)*

Description

Calculates Heat Index (HI) from air temperature and relative humidity. Handles input unit declaration via argument and/or `attr(x, "unit")`. The C++ core computes HI in Fahrenheit; output can be returned in `"degF"` or `"degC"`.

Usage

```
calcHI(
  airTemp,
  relativeHumidity,
  inputunits = NULL,
  outputunits = "degF",
  roundby = 1,
  returnWithUnits = TRUE,
  ignoreattr = FALSE,
  debug = FALSE
)
```

Arguments

`airTemp`	Numeric vector of air temperature. May carry `attr(., "unit")`.
`relativeHumidity`	Numeric vector of RH in percent (0–100).
`inputunits`	Character or `NULL`; one of `"degC"`, `"degF"`, `"K"`. Only needed if `ignoreattr = TRUE` and no attribute is present.
`outputunits`	Character; `"degF"` (default) or `"degC"` for the returned HI.
`roundby`	Integer; decimal places to round the output (default `1`).
`returnWithUnits`	Logical; if `TRUE`, sets `attr(result, "unit")` to `outputunits`.
`ignoreattr`	Logical; see Unit handling.
`debug`	Logical; if `TRUE`, emit a brief trace of unit decisions.

Value

Numeric vector of Heat Index in `outputunits`. If `returnWithUnits = TRUE`, the result has `attr(x, "unit") = outputunits`.

Unit handling

- If `ignoreattr = FALSE` (default), the function requires `attr(airTemp, "unit")` and enforces agreement with `inputunits` when provided; otherwise it errors on mismatch/missing attribute.
- If `ignoreattr = TRUE`, the function accepts either the attribute or `inputunits` (but needs at least one).
- Internally, temperature is normalized to `"degF"` before calling the C++ core.

See Also

`unit` for lightweight unit tagging.

Examples

```
# Attribute-driven (degC -> converted internally to degF for core)
ta <- c(30, 35, 40); attr(ta, "unit") <- "degC"
rh <- c(50, 60, 65)
hiF <- calcHI(ta, rh, outputunits = "degF")
attr(hiF, "unit")  # "degF"
```

```
# Explicit input units, Celsius output
hiC <- calcHI(ta, rh, inputunits = "degC", outputunits = "degC")

# Ignore attribute and trust supplied units
attr(ta, "unit") <- "degF"  # wrong on purpose
hi_ok <- calcHI(ta, rh, inputunits = "degC", outputunits = "degF", ignoreattr = TRUE)

# data.table pattern
if (requireNamespace("data.table", quietly = TRUE)) {
  DT <- data.table::data.table(ta = c(30, 35), rh = c(50, 60))
  attr(DT$ta, "unit") <- "degC"
  DT[, hi := calcHI(ta, rh, outputunits = "degC")]
  attr(DT$hi, "unit")  # "degC"
}
```

`calcPres` *Sea-level pressure from station pressure*

Description

Converts station pressure (in millibars / hPa) to sea-level pressure using a standard atmosphere approximation and the station elevation.

Usage

```
calcPres(
  pressureMB,
  airTemp,
  elevation,
  inputunits = "degC",
  elevUnits = "m",
  ignoreattr = TRUE,
  quiet = TRUE
)
```

Arguments

`pressureMB`	Numeric vector. Station pressure in millibars (hPa).
`airTemp`	Numeric vector. Air temperature at the station. Units set by `inputunits`.
`elevation`	Numeric vector or length-1 scalar. Station elevation. Units set by `elevUnits`.
`inputunits`	Character scalar. Temperature units: `"degC"` (default) or `"degF"`.
`elevUnits`	Character scalar. Elevation units: `"m"` (default) or `"ft"`.
`ignoreattr`	Logical.
`quiet`	Logical.

Details

If `elevation` is a scalar, it is recycled to match `pressureMB`.

Value

Numeric vector of sea-level pressure (mb / hPa), same length as `pressureMB`.

Examples

```
# Example: convert 1000 mb at 20C and 100 m elevation
calcPres(pressureMB = 1000, airTemp = 20, elevation = 100)

# Vectorized
calcPres(pressureMB = c(995, 1002), airTemp = c(15, 18), elevation = 120)
```

`calcRH` *Compute Relative Humidity from Temperature and Dew Point (or VPD)*

Description

Computes RH (%) given air temperature and dew point, or alternatively air temperature and vapor pressure deficit (VPD, kPa). Handles temperature unit declaration via argument and/or `attr(x, "unit")`.

Usage

```
calcRH(
  airTemp,
  dewPoint = NULL,
  inputunits = "degC",
  vpd = NULL,
  ignoreattr = FALSE,
  debug = FALSE
)
```

Arguments

`airTemp`	Numeric vector of air temperature. May carry `attr(., "unit")`.
`dewPoint`	Numeric vector of dew point temperature (optional if `vpd` is supplied). May carry `attr(., "unit")`.
`inputunits`	Character; temperature unit token for `airTemp`/`dewPoint`. One of `"degC"`, `"degF"`, `"K"` (also accepts `"C"`, `"F"`). Defaults to `"degC"`.
`vpd`	Numeric vector of vapor pressure deficit in kPa. If provided, `dewPoint` is ignored.
`ignoreattr`	Logical; if `TRUE`, relax attribute checks and trust `inputunits`.
`debug`	Logical; if `TRUE`, emit a brief trace of unit decisions.

Details

Internally, temperatures are normalized to `"degC"` or `"degF"` depending on `inputunits`, then passed to a C++ routine. When `vpd` is supplied, `dewPoint` is ignored and RH is computed from `airTemp` and `vpd`.

Value

Numeric vector of relative humidity in percent (0–100). No unit attribute is attached.

Unit handling

- Recognized temperature tokens: `"degC"`, `"degF"`, `"K"`.
- `inputunits` applies to `airTemp` (and `dewPoint` if provided).
- If `ignoreattr = FALSE` (default), the function requires unit attributes on provided temperature vectors and enforces agreement with `inputunits` when specified.
- If `inputunits == "K"`, values are normalized to `"degC"` for the core.

See Also

`calcTD`, `unit`.

Examples

```
# Using dew point
ta <- c(30, 31); dp <- c(20, 21)
attr(ta, "unit") <- "degC"; attr(dp, "unit") <- "degC"
rh <- calcRH(ta, dp, inputunits = "degC")
rh  # percent

# Using VPD (kPa), ignoring attributes and declaring units
taF <- c(86, 88); vpd <- c(1.5, 2.0)
rh2 <- calcRH(taF, vpd = vpd, inputunits = "degF", ignoreattr = TRUE)

# data.table pattern
if (requireNamespace("data.table", quietly = TRUE)) {
  DT <- data.table::data.table(ta = ta, dp = dp)
  attr(DT$ta, "unit") <- "degC"; attr(DT$dp, "unit") <- "degC"
  DT[, rh := calcRH(ta, dewPoint = dp, inputunits = "degC")]
}
```

---

calcTD *Compute Dew Point Temperature (fast C++ core)*

Description

Calculates dew point from air temperature and relative humidity using a C++ implementation. Inputs can be declared via `inputunits` and/or inferred from `attr(x, "unit")` on `airTemp`. Output is returned in the requested unit and, by default, tagged with a lightweight `"unit"` attribute.

Usage

```
calcTD(
  airTemp,
  relativeHumidity,
  inputunits = "degC",
  outputunits = "degC",
  roundby = 2,
  returnWithUnits = TRUE,
  ignoreattr = FALSE,
  debug = FALSE
)
```

Arguments

`airTemp`	Numeric vector of air temperature. May carry `attr(., "unit")`.
`relativeHumidity`	Numeric vector of RH in percent (0–100).
`inputunits`	Character; temperature units of `airTemp`. One of `"degC"`, `"degF"`, `"K"`. Defaults to `"degC"`.
`outputunits`	Character; desired dew point units. One of `"degC"`, `"degF"`. Defaults to `"degC"`.
`roundby`	Integer; number of decimal places to round the result (performed in C++).
`returnWithUnits`	Logical; if `TRUE`, sets `attr(result, "unit")` to `outputunits`.
`ignoreattr`	Logical; if `TRUE`, skip attribute checks and rely on `inputunits`.
`debug`	Logical; if `TRUE`, emit a brief trace of unit decisions.

Value

Numeric vector of dew point in `outputunits`. If `returnWithUnits` = `TRUE`, the result has `attr(x, "unit")` = `outputunits`.

Unit handling

- Recognized temperature tokens: `"degC"`, `"degF"`, `"K"`.
- If `ignoreattr` = `FALSE` (default), the function reads `attr(airTemp, "unit")` and enforces agreement with `inputunits` when provided; otherwise it errors on conflict or missing attribute.
- If `ignoreattr` = `TRUE`, the function trusts `inputunits` (or the attribute if `inputunits` is missing).
- Internally, the C++ core expects either `"degC"` or `"degF"`. If `inputunits` == `"K"`, values are normalized to `"degC"`.

See Also

`unit` for lightweight unit tagging.

Examples

```
# Simple vector (attribute-driven)
ta <- c(30, 31, 29); attr(ta, "unit") <- "degC"
rh <- c(50, 55, 60)
dpC <- calcTD(ta, rh, outputunits = "degC")
attr(dpC, "unit")  # "degC"

# Override: declare input as Fahrenheit, request Fahrenheit output
taF <- c(86, 88, 84); attr(taF, "unit") <- "degF"
dpF <- calcTD(taF, rh, inputunits = "degF", outputunits = "degF")
```

```
# Ignore attribute and trust inputunits
attr(taF, "unit") <- "degC"  # wrong on purpose
dp_ok <- calcTD(taF, rh, inputunits = "degF", outputunits = "degC", ignoreattr = TRUE)

# data.table pattern
if (requireNamespace("data.table", quietly = TRUE)) {
  DT <- data.table::data.table(ta = ta, rh = rh)
  attr(DT$ta, "unit") <- "degC"
  DT[, dp := calcTD(ta, rh, outputunits = "degC")]
  attr(DT$dp, "unit")
}
```

calcWB *Calculate Wet-Bulb Temperature (strict, attribute-aware)*

Description

Computes wet-bulb temperature (WB) from air temperature and relative humidity, using a compiled C++ routine (`calcWB_cpp`). This version follows your lightweight unit model:

- Inputs are plain numerics with an optional `attr(x, "unit")`.
- By default (`ignoreattr = FALSE`) a **unit attribute is required** on `airTemp` (and must match `inputunits` if you also supply it). Set `ignoreattr = TRUE` to skip attribute checks and rely solely on `inputunits`.
- Accepts temperature units `"degC"`, `"degF"`, or `"K"`. When `"K"` is used, values are converted to `"degC"` internally for the C++ call.

`relativeHumidity` may be given as **percent** (0–100) or **fraction** (0–1). Fractions (<= 1.5) are auto-scaled to percent.

Usage

```
calcWB(
  airTemp,
  relativeHumidity,
  inputunits = "degF",
  outputunits = "degF",
  method = NULL,
  ignoreattr = FALSE,
  returnWithUnits = TRUE,
  debug = FALSE
)
```

Arguments

`airTemp`	Numeric. Air temperature. Provide `attr(airTemp, "unit")` as `"degC"`, `"degF"`, or `"K"` unless `ignoreattr = TRUE`.
`relativeHumidity`	Numeric. Relative humidity; percent (0–100) or fraction (0–1). Fractions are auto-converted to percent.

inputunits	Character. Temperature unit of airTemp ("degF", "degC", or "K"). Required when ignoreattr = TRUE; if provided together with an attribute, they must agree. Default: "degF".
outputunits	Character. Desired WB output unit ("degF" or "degC"). Default: "degF".
method	Character or NULL. Passed through to calcWB_cpp to select an algorithmic variant (keep NULL for the default).
ignoreattr	Logical. If FALSE (default), require a unit attribute on airTemp and validate against inputunits when provided. If TRUE, skip attribute checks and rely only on inputunits.
returnWithUnits	Logical. If TRUE, tag the numeric result with attr(, "unit") = outputunits. Default: FALSE.
debug	Logical. If TRUE, print a compact summary of normalized inputs.

Value

Numeric vector of wet-bulb temperature in outputunits. If returnWithUnits = TRUE, the numeric is tagged with attr(, "unit").

Examples

```
## Not run:
# Strict mode (attribute required):
T <- 90; attr(T, "unit") <- "degF"
calcWB(T, 70)                        # returns degF by default
calcWB(T, 70, outputunits = "degC")  # convert to degC

# Kelvin input:
Tk <- 305.15; attr(Tk, "unit") <- "K"
calcWB(Tk, 0.6, outputunits = "degC")                # RH as fraction → auto scaled

# Ignore attributes and use explicit units:
calcWB(32, 40, inputunits = "degC", ignoreattr = TRUE)

## End(Not run)
```

---

calcWindchill *Wind chill (US NWS formula)*

Description

Computes wind chill using the standard U.S. National Weather Service formula.

Usage

```
calcWindchill(airTemp, sfcWind, roundBy = 1)
```

Arguments

airTemp	Numeric. Air temperature in degrees Fahrenheit.
sfcWind	Numeric. Wind speed in mph.
roundBy	Integer. Number of decimal places to round to (default 1).

Value

Numeric vector of wind chill values.

Examples

```
calcWindchill(airTemp = 30, sfcWind = 10)
calcWindchill(airTemp = c(30, 25), sfcWind = c(10, 15), roundBy = 0)
```

`unit` *Lightweight unit getter/setter (conversion + optional rounding)*

Description

Provides a simple way to read and set a `"unit"` attribute on numeric vectors. The replacement form (`unit(x) <- value`) can also convert values between supported units (handled by the package-internal `.convert_units()`) and optionally round the result.

Usage

```
unit(x)

unit(x) <- value
```

Arguments

`x`	A numeric vector (or column) to query or tag with a `"unit"` attribute.
`value`	Character scalar describing the assignment. See the accepted forms above. Examples: `"degF"`, `"degC -> degF"`, `"degC|degF|1"`, `"degF (2)"`, `"degF; round=1"`.

Details

Accepted assignment forms for `value`:

- `"degF"` — set/convert to `degF` using the current attribute as the source (if present).
- `"degC -> degF"` — explicitly convert from `degC` to `degF`.
- `"degC|degF"` — shorthand for the arrow form above.
- Optional rounding may be requested (highest precedence first): `"...; round=1"`, `"...; digits=1"`, trailing `"|1"`, or trailing `"(1)"`; e.g., `"degC|degF|1"` or `"degF (2)"`.

If an explicit source unit is provided on the left side (e.g., `"degC -> degF"`), it is treated as authoritative. To prevent accidental re-interpretation of already-tagged data, a source-mismatch policy is applied:

- `options(jj.unit.on_src_mismatch = "warn_noop")` (default) — warn and do nothing when *declared source* differs from `attr(x,"unit")`.
- `"error"` — stop with an error.
- `"convert"` — trust the declared source and convert anyway (original permissive behavior).

Rounding is applied *after* conversion. Global default rounding can be set via `options(jj.unit.round_digits = K)`. This default is only applied when a conversion actually occurred; inline digits (`round=` / `digits=` / `|K` / `(K)`) always apply. When rounding occurs, the setter also records `attr(x, "unit_digits") = K`.

To avoid silently mislabeling temperatures, you may forbid "tag-only" temperature assignments (no known source) by enabling: `options(jj.unit.disallow_temp_tag_only = TRUE)`. With this option set, attempting `unit(x) <- "degF"` on an untagged vector will error; use `"src|dst"` instead.

Value

- `unit(x)` returns the current unit attribute (character scalar) or `NULL`.
- `unit(x) <- value` returns the modified vector `x`, with values converted if needed, optionally rounded, and `attr(x, "unit")` set to the target unit. When rounding is applied, `attr(x, "unit_digits")` is also set.

Conversions

The actual numeric conversions are performed by the internal `.convert_units(x, from, to)`. Typical pairs supported in this package include temperature (`degC`, `degF`, `K`/`degK`), pressure (`hPa`, `Pa`), wind speed (`m/s`, `mph`, `kt`), precipitation depth (`kg/m^2`, `mm`, `in`), radiation (`W/m^2`), and relative humidity forms (where supported by your map).

Examples

```
x <- c(25, 26, 27)
unit(x) <- "degC"        # tag as degC
unit(x)                  # "degC"

# Convert using current attribute as the source
unit(x) <- "degF"        # C -> F (if attr is "degC")

# Explicit source -> target
unit(x) <- "degF -> degC"

# Shorthand with rounding
unit(x) <- "degC|degF|1"   # C -> F, then round(., 1)
unit(x) <- "degF (2)"      # tag/convert to F, then round(., 2)

# Safer everyday pattern (idempotent):
y <- c(0, 5, 10); unit(y) <- "degC"
unit(y) <- "degF"          # converts once; calling again is a no-op

# data.table in-place usage
if (requireNamespace("data.table", quietly = TRUE)) {
  library(data.table)
  DT <- data.table(ta = c(25, 26, 27))
  unit(DT$ta) <- "degC"
  DT[, ta := { unit(ta) <- "degC -> degF; round=1"; ta }]
  unit(DT$ta)                    # "degF"
  attr(DT$ta, "unit_digits")     # 1
}

# Policies (optional):
# options(jj.unit.on_src_mismatch = "warn_noop")  # default
# options(jj.unit.on_src_mismatch = "error")
# options(jj.unit.on_src_mismatch = "convert")
#
# options(jj.unit.round_digits = 1L)             # global rounding (after conversions)
# options(jj.unit.disallow_temp_tag_only = TRUE)  # forbid tag-only for temperatures
```

---

unitConvertRound *Convert and Round a Vector Between Units*

Description

convert_units_round converts numeric vectors from one unit to another using the **units** package, then rounds the result to a specified number of decimal places. This can handle temperature (e.g., "degC" $\rightarrow$ "degF"), speed ("m/s" $\rightarrow$ "mi/h"), and any other convertible units supported by **units**.

Usage

```
unitConvertRound(x, from, to, digits = 2, strip = FALSE)
```

Arguments

x	Numeric or units object. The values to convert.
from	Character. Original unit of x (e.g., "degC", "m/s").
to	Character. Desired target unit (e.g., "degF", "mi/h").
digits	Integer. Number of decimal places to round the converted values (default 1).

Details

Internally, this function wraps set_units(x, from) and set_units(..., to), then drops the units and applies rounding. It requires that the **units** package be installed and that the requested units are compatible for conversion.

Value

A numeric vector of the converted and rounded values.

Examples

```
## Not run:
# Convert WBGT from °C to °F
wbgt_c <- c(20, 25, 30)
unitConvertRound(wbgt_c, from = "degC", to = "degF", digits = 1)

# Convert wind speed from m/s to mi/h
speed_ms <- c(5, 10, 15)
unitConvertRound(speed_ms, from = "m/s", to = "mi/h", digits = 2)

## End(Not run)
```

uv2wdws *Convert u/v wind components to direction and speed*

Description

Converts u (zonal, positive eastward) and v (meridional, positive northward) wind components into meteorological wind direction (degrees from which the wind is blowing) and wind speed.

Usage

```
uv2wdws(u, v)
```

Arguments

`u`	Numeric vector. Zonal wind component.
`v`	Numeric vector. Meridional wind component.

Value

A numeric matrix with two columns: `wd` (wind direction degrees, [0, 360)) and `ws` (wind speed, same units as `u/v`).

Examples

```
uv2wdws(u = c(1, 0, -1), v = c(0, 1, 0))
```

winddeg *Wind direction degrees to 16-point compass labels*

Description

Converts wind direction degrees into standard 16-point compass labels: N, NNE, NE, ENE, E, ESE, SE, SSE, S, SSW, SW, WSW, W, WNW, NW, NNW.

Usage

```
winddeg(windDeg)
```

Arguments

`windDeg`	Numeric vector of wind directions in degrees.

Value

Character vector of compass labels (same length as `windDeg`).

Examples

```
winddeg(c(0, 20, 45, 90, 200, 359.9))
```

Index