

Package ‘bigleaf’

May 1, 2022

Type Package

Version 0.8.1

Date 2022-04-27

Title Physical and Physiological Ecosystem Properties from Eddy
Covariance Data

Maintainer Juergen Knauer <J.Knauer@westernsydney.edu.au>

Description Calculation of physical (e.g. aerodynamic conductance, surface temperature),
and physiological (e.g. canopy conductance, water-use efficiency) ecosystem properties
from eddy covariance data and accompanying meteorological measurements. Calculations
assume the land surface to behave like a 'big-
leaf' and return bulk ecosystem/canopy variables.

URL <https://bitbucket.org/juergenknauer/bigleaf>

BugReports <https://bitbucket.org/juergenknauer/bigleaf/issues>

Depends R (>= 2.10)

Imports robustbase, solartime

License GPL (>= 2)

Encoding UTF-8

LazyData yes

RoxygenNote 7.1.2

Suggests knitr, rmarkdown, testthat

VignetteBuilder knitr

NeedsCompilation no

Author Juergen Knauer [aut, cre] (<<https://orcid.org/0000-0002-4947-7067>>),
Thomas Wutzler [aut] (<<https://orcid.org/0000-0003-4159-5445>>),
Alison Bennett [aut] (<<https://orcid.org/0000-0002-8249-976X>>),
Soenke Zaehle [ctb] (<<https://orcid.org/0000-0001-5602-7956>>),
Tarek El-Madany [ctb] (<<https://orcid.org/0000-0002-0726-7141>>),
Mirco Migliavacca [ctb] (<<https://orcid.org/0000-0003-3546-8407>>)

Repository CRAN

Date/Publication 2022-05-01 14:30:05 UTC

R topics documented:

aerodynamic.conductance	3
air.density	8
Arrhenius.temp.response	9
AT_Neu_Jul_2010	10
bigleaf.constants	11
biochemical.energy	13
decoupling	14
dew.point	16
DE_Tha_Jun_2014	18
energy.closure	19
energy.use.efficiency	21
equilibrium.imposed.ET	22
Esat.slope	24
extraterrestrial.radiation	25
filter.data	26
filter.growing.season	29
FR_Pue_May_2012	30
Gb.Choudhury	31
Gb.Su	34
Gb.Thom	37
intercellular.CO2	38
isothermal.Rn	40
kg.to.mol	41
kinematic.viscosity	42
latent.heat.vaporization	43
LE.to.ET	44
light.response	45
light.use.efficiency	46
longwave.conductance	47
Monin.Obukhov.length	48
ms.to.mol	49
optimum.temperature	50
photosynthetic.capacity	52
potential.ET	56
potential.radiation	59
pressure.from.elevation	60
psychrometric.constant	61
radiometric.surface.temp	62
reference.ET	63
Reynolds.Number	64
Rg.to.PPFD	65
roughness.length.heat	66
roughness.parameters	67
stability.correction	70
stability.parameter	71
stomatal.sensitivity	72

stomatal.slope	74
surface.CO2	77
surface.conditions	78
surface.conductance	80
umolCO2.to.gC	83
virtual.temp	84
VPD.to.rH	85
wetbulb.temp	87
wind.profile	88
WUE.metrics	90

Index 93

aerodynamic.conductance

Aerodynamic Conductance

Description

Bulk aerodynamic conductance, including options for the boundary layer conductance formulation and stability correction functions.

Usage

```
aerodynamic.conductance(
  data,
  Tair = "Tair",
  pressure = "pressure",
  wind = "wind",
  ustar = "ustar",
  H = "H",
  zr,
  zh,
  d,
  z0m = NULL,
  D1,
  N = 2,
  fc = NULL,
  LAI,
  Cd = 0.2,
  hs = 0.01,
  wind_profile = FALSE,
  stab_correction = TRUE,
  stab_formulation = c("Dyer_1970", "Businger_1971"),
  Rb_model = c("Thom_1972", "Choudhury_1988", "Su_2001", "constant_kB-1"),
  kB_h = NULL,
  Sc = NULL,
  Sc_name = NULL,
```

```

    constants = bigleaf.constants()
)

```

Arguments

data	Data.frame or matrix containing all required variables
Tair	Air temperature (deg C)
pressure	Atmospheric pressure (kPa)
wind	Wind speed (m s-1)
ustar	Friction velocity (m s-1)
H	Sensible heat flux (W m-2)
zr	Instrument (reference) height (m)
zh	Canopy height (m)
d	Zero-plane displacement height (m)
z0m	Roughness length for momentum (m), optional; if not provided, it is estimated from roughness.parameters (method="wind_profile"). Only used if wind_profile = TRUE and/or Rb_model = "Su_2001" or "Choudhury_1988".
Dl	Characteristic leaf dimension (m) (if Rb_model = "Su_2001") or leaf width (if Rb_model = "Choudhury_1988"); ignored otherwise.
N	Number of leaf sides participating in heat exchange (1 or 2); only used if Rb_model = "Su_2001". Defaults to 2.
fc	Fractional vegetation cover (-); only used if Rb_model = "Su_2001". See Details.
LAI	One-sided leaf area index (m ² m-2); only used if Rb_model = "Choudhury_1988" or "Su_2001".
Cd	Foliage drag coefficient (-); only used if Rb_model = "Su_2001".
hs	Roughness length of bare soil (m); only used if Rb_model = "Su_2001".
wind_profile	Should Ga for momentum be calculated based on the logarithmic wind profile equation? Defaults to FALSE.
stab_correction	Should stability correction be applied? Defaults to TRUE. Ignored if wind_profile = FALSE.
stab_formulation	Stability correction function. Either "Dyer_1970" (default) or "Businger_1971". Ignored if wind_profile = FALSE or if stab_correction = FALSE.
Rb_model	Boundary layer resistance formulation. One of "Thom_1972", "Choudhury_1988", "Su_2001", "constant_kB-1"
kB_h	kB-1 value for heat transfer; only used if Rb_model = "constant_kB-1"
Sc	Optional: Schmidt number of additional quantities to be calculated
Sc_name	Optional: Name of the additional quantities, has to be of same length than Sc_name

constants	k - von Karman constant
	cp - specific heat of air for constant pressure (J K ⁻¹ kg ⁻¹)
	Kelvin - conversion degree Celsius to Kelvin
	g - gravitational acceleration (m s ⁻²)
	pressure0 - reference atmospheric pressure at sea level (Pa)
	Tair0 - reference air temperature (K)
	Sc_CO2 - Schmidt number for CO2
	Pr - Prandtl number (if Sc is provided)

Details

Aerodynamic conductance for heat (G_{a_h}) is calculated as:

$$G_{a_h} = 1/(Ra_m + Rb_h)$$

where Ra_m is the aerodynamic resistance for momentum and Rb_h the (quasi-laminar) canopy boundary layer resistance ('excess resistance').

The aerodynamic resistance for momentum Ra_m is given by:

$$Ra_m = u/ustar^2$$

Note that this formulation accounts for changes in atmospheric stability, and does not require an additional stability correction function.

An alternative method to calculate Ra_m is provided (calculated if `wind_profile = TRUE`):

$$Ra_m = (\ln((zr - d)/z0m) - psi_h)/(kustar)$$

If the roughness parameters $z0m$ and d are unknown, they can be estimated using [roughness.parameters](#).

The argument `stab_formulation` determines the stability correction function used to account for the effect of atmospheric stability on Ra_m (Ra_m is lower for unstable and higher for stable stratification). Stratification is based on a stability parameter zeta ($(z-d)/L$), where z = reference height, d the zero-plane displacement height, and L the Monin-Obukhov length, calculated with [Monin.Obukhov.length](#). The stability correction function is chosen by the argument `stab_formulation`. Options are "Dyer_1970" and "Businger_1971".

The model used to determine the canopy boundary layer resistance for heat (Rb_h) is specified by the argument `Rb_model`. The following options are implemented: "Thom_1972" is an empirical formulation based on the friction velocity ($ustar$) (Thom 1972):

$$Rb_h = 6.2ustar^{-0.667}$$

The model by Choudhury & Monteith 1988 (`Rb_model = "Choudhury_1988"`), calculates Rb_h based on leaf width, LAI and $ustar$ (Note that function argument `Dl` represents leaf width (w) and not characteristic leaf dimension (DI) if `Rb_model = "Choudhury_1988"`):

$$G_{b_h} = LAI((0.02/\alpha) * \text{sqrt}(u(zh)/w) * (1 - \exp(-\alpha/2)))$$

where α is a canopy attenuation coefficient modeled in dependence on LAI, $u(zh)$ is wind speed at canopy height (calculated from `wind.profile`), and w is leaf width (m). See [Gb.Choudhury](#) for further details.

The option `Rb_model = "Su_2001"` calculates `Rb_h` based on the physically-based `Rb` model by Su et al. 2001, a simplification of the model developed by Massman 1999:

$$kB_h = (kCd\alpha^2)/(4Ctustar/u(zh)) + kB_s - 1(1 - fc)^2$$

where Cd is a foliage drag coefficient (defaults to 0.2), fc is fractional vegetation cover, $Bs-1$ is the inverse Stanton number for bare soil surface, and Ct is a heat transfer coefficient. See [Gb.Su](#) for details on the model.

The models calculate the parameter kB^{-1} (in the code referred to as `kB_h`), which is related to `Rb_h`:

$$kB_h = Rb_h * (k * ustar)$$

From version 0.7.6 onwards, the roughness length for heat (`z0h`) is added to the output if `z0m` is available (i.e. provided as input or calculated within this function). `z0h` is calculated from [roughness.length.heat](#):

$$z0h = z0m / \exp(kB_h)$$

`Rb` (and `Gb`) for water vapor and heat are assumed to be equal in this package. `Gb` for other quantities x is calculated as (Hicks et al. 1987):

$$Gb_x = Gb / (Sc_x / Pr)^{0.67}$$

where Sc_x is the Schmidt number of quantity x , and Pr is the Prandtl number (0.71).

Value

a `data.frame` with the following columns:

<code>Ga_m</code>	Aerodynamic conductance for momentum transfer (m s-1)
<code>Ra_m</code>	Aerodynamic resistance for momentum transfer (s m-1)
<code>Ga_h</code>	Aerodynamic conductance for heat transfer (m s-1)
<code>Ra_h</code>	Aerodynamic resistance for heat transfer (s m-1)
<code>Gb_h</code>	Canopy boundary layer conductance for heat transfer (m s-1)
<code>Rb_h</code>	Canopy boundary layer resistance for heat transfer (s m-1)
<code>kB_h</code>	kB^{-1} parameter for heat transfer
<code>z0h</code>	Roughness length for heat (m) (NA if not input <code>z0m</code> not provided as input or not estimated in this function)
<code>zeta</code>	Stability parameter 'zeta' (NA if <code>wind_profile = FALSE</code>)
<code>psi_h</code>	Integrated stability correction function (NA if <code>wind_profile = FALSE</code>)

Ra_CO2	Aerodynamic resistance for CO2 transfer (s m-1)
Ga_CO2	Aerodynamic conductance for CO2 transfer (m s-1)
Gb_CO2	Canopy boundary layer conductance for CO2 transfer (m s-1)
Ga_Sc_name	Aerodynamic conductance for Sc_name (m s-1). Only added if Sc_name and the respective Sc are provided
Gb_Sc_name	Boundary layer conductance for Sc_name (m s-1). Only added if Sc_name and the respective Sc are provided

Note

Input variables such as LAI, DI, or zh can be either constants, or vary with time (i.e. vectors of the same length as data).

Note that boundary layer conductance to water vapor transfer (Gb_w) is often assumed to equal Gb_h. This assumption is also made in this R package, for example in the function [surface.conductance](#).

If the roughness length for momentum (z0m) is not provided as input, it is estimated from the function `roughness.parameters` within `wind.profile` if `wind.profile = TRUE` and/or `Rb_model = "Su_2001"` or `"Choudhury_1988"`. The `roughness.parameters` function estimates a single z0m value for the entire time period! If a varying z0m value (e.g. across seasons or years) is required, z0m should be provided as input argument.

References

Verma, S., 1989: Aerodynamic resistances to transfers of heat, mass and momentum. In: Estimation of areal evapotranspiration, IAHS Pub, 177, 13-20.

Verhoef, A., De Bruin, H., Van Den Hurk, B., 1997: Some practical notes on the parameter kB-1 for sparse vegetation. Journal of Applied Meteorology, 36, 560-572.

Hicks, B.B., Baldocchi, D.D., Meyers, T.P., Hosker, J.R., Matt, D.R., 1987: A preliminary multiple resistance routine for deriving dry deposition velocities from measured quantities. Water, Air, and Soil Pollution 36, 311-330.

Monteith, J.L., Unsworth, M.H., 2008: Principles of environmental physics. Third Edition. Elsevier Academic Press, Burlington, USA.

See Also

[Gb.Thom](#), [Gb.Choudhury](#), [Gb.Su](#) for calculations of Rb / Gb only

Examples

```
df <- data.frame(Tair=25,pressure=100,wind=c(3,4,5),ustar=c(0.5,0.6,0.65),H=c(200,230,250))

# simple calculation of Ga
aerodynamic.conductance(df,Rb_model="Thom_1972")

# calculation of Ga using a model derived from the logarithmic wind profile
aerodynamic.conductance(df,Rb_model="Thom_1972",zr=40,zh=25,d=17.5,z0m=2,wind_profile=TRUE)

# simple calculation of Ga_m, but a physically based canopy boundary layer model
```

```
aerodynamic.conductance(df,Rb_model="Su_2001",zr=40,zh=25,d=17.5,Dl=0.05,N=2,fc=0.8)
```

```
air.density
```

```
Air Density
```

Description

Air density of moist air from air temperature and pressure.

Usage

```
air.density(Tair, pressure, constants = bigleaf.constants())
```

Arguments

Tair	Air temperature (deg C)
pressure	Atmospheric pressure (kPa)
constants	Kelvin - conversion degC to Kelvin Rd - gas constant of dry air (J kg-1 K-1) kPa2Pa - conversion kilopascal (kPa) to pascal (Pa)

Details

Air density (ρ) is calculated as:

$$\rho = \text{pressure} / (\text{Rd} * \text{Tair})$$

Value

ρ air density (kg m-3)

References

Foken, T, 2008: Micrometeorology. Springer, Berlin, Germany.

Examples

```
# air density at 25degC and standard pressure (101.325kPa)
air.density(25,101.325)
```

 Arrhenius.temp.response

(Modified) Arrhenius Temperature Response Function

Description

(Modified) Arrhenius function describing the temperature response of biochemical parameters.

Usage

```

Arrhenius.temp.response(
  param,
  Temp,
  Ha,
  Hd,
  dS,
  constants = bigleaf.constants()
)

```

Arguments

param	Parameter measured at measurement temperature (umol m ⁻² s ⁻¹)
Temp	Measurement temperature (degC)
Ha	Activation energy for param (kJ mol ⁻¹)
Hd	Deactivation energy for param (kJ mol ⁻¹)
dS	Entropy term for param (kJ mol ⁻¹ K ⁻¹)
constants	Kelvin - conversion degree Celsius to Kelvin Rgas - universal gas constant (J mol ⁻¹ K ⁻¹) kJ2J - conversion kilojoule (kJ) to joule (J)

Details

The function returns the biochemical rate at a reference temperature of 25degC given a predefined temperature response function. This temperature response is given by a modified form of the Arrhenius function:

$$param_{25} = param / (exp(Ha * (Temp - Tref) / (Tref * Rgas * Temp)) * (1 + exp((Tref * dS - Hd) / (Tref * Rgas)))) / (1 +$$

where param is the value/rate of the parameter at measurement temperature, Temp is temperature in K, Tref is reference temperature (298.15K), and Rgas is the universal gas constant (8.314 J K⁻¹ mol⁻¹). Ha is the activation energy (kJ mol⁻¹), Hd is the deactivation energy (kJ mol⁻¹), and dS is the entropy term (kJ mol⁻¹ K⁻¹) of the respective parameter.

If either Hd or dS or both are not provided, the equation above reduces to the first term (i.e. the common Arrhenius equation without the deactivation term.)

Value

param25 - value of the input parameter at the reference temperature of 25degC (umol m⁻² s⁻¹)

References

Johnson F.H., Eyring H., Williams R.W. 1942: The nature of enzyme inhibitions in bacterial luminescence: sulfanilamide, urethane, temperature and pressure. *Journal of cellular and comparative physiology* 20, 247-268.

Kattge J., Knorr W., 2007: Temperature acclimation in a biochemical model of photosynthesis: a reanalysis of data from 36 species. *Plant, Cell and Environment* 30, 1176-1190.

 AT_Neu_Jul_2010

Eddy Covariance Data of AT-Neu (Neustift)

Description

Halfhourly eddy covariance Data of the site AT-Neu, a mountain meadow in Austria. (<https://sites.fluxdata.org/AT-Neu/>). Data are from July 2010.

Usage

AT_Neu_Jul_2010

Format

A data frame with 1488 observations and 31 columns:

year year of measurement

month month of measurement

doy day of year

hour hour (0 - 23.5)

Tair Air temperature (degC) [TA_F]

Tair_qc Quality control of Tair [TA_F_QC]

PPFD Photosynthetic photon flux density (umol m⁻² s⁻¹) [PPFD_IN]

PPFD_qc Quality control of PPFD [PPFD_IN_QC]

VPD Vapor pressure deficit (kPa) [VPD_F]

VPD_qc Quality control of VPD [VPD_F_QC]

pressure Atmospheric pressure (kPa) [PA_F]

precip precipitation (mm) [P_F]

precip_qc Quality control of precip [P_F_QC]

ustar friction velocity (m s⁻¹) [USTAR]

wind horizontal wind velocity (m s⁻¹) [WS_F]

wind_qc Quality control of wind [WS_F_QC]
Ca CO2 concentration (ppm) [CO2_F_MDS]
Ca_qc Quality control of Ca [CO2_F_MDS_QC]
LW_up upward longwave radiation (W m-2) [LW_OUT]
Rn Net radiation (W m-2) [NETRAD]
LE Latent heat flux (W m-2) [LE_F_MDS]
LE_qc Quality control of LE [LE_F_MDS_QC]
H Sensible heat flux (W m-2) [H_F_MDS]
H_qc Quality control of H [H_F_MDS_QC]
G Ground heat flux (W m-2) [G_F_MDS]
G_qc Quality control of G [G_F_MDS_QC]
NEE Net ecosystem exchange (umol m-2 s-1) [NEE_VUT_USTAR50]
NEE_qc Quality control of NEE [NEE_VUT_USTAR50_QC]
GPP Gross primary productivity from nighttime partitioning (umol m-2 s-1) [GPP_NT_VUT_USTAR50]
GPP_qc Quality control of GPP [NEE_VUT_USTAR50_QC]
Reco Ecosystem respiration from nighttime partitioning (umol m-2 s-1) [RECO_NT_VUT_USTAR50]

Note

The original variable names as provided by the FLUXNET2015 dataset are given in squared brackets. Note that variable units have been converted in some cases (e.g. VPD from hPa to kPa).

Source

original data were downloaded from <https://fluxnet.org/> (accessed 09 November 2016)

bigleaf.constants *Constants Used in the bigleaf Package*

Description

This function defines the following constants:

Usage

```

bigleaf.constants(
  cp = 1004.834,
  Rgas = 8.31451,
  Rv = 461.5,
  Rd = 287.0586,
  Md = 0.0289645,
  Mw = 0.0180153,
  eps = 0.622,

```

```

g = 9.81,
solar_constant = 1366.1,
pressure0 = 101325,
Tair0 = 273.15,
k = 0.41,
Cmol = 0.012011,
Omol = 0.0159994,
H2Omol = 0.01801528,
sigma = 5.670367e-08,
Pr = 0.71,
Sc_CO2 = 1.07,
Le067 = 0.93,
Kelvin = 273.15,
DwDc = 1.6,
days2seconds = 86400,
kPa2Pa = 1000,
Pa2kPa = 0.001,
umol2mol = 1e-06,
mol2umol = 1e+06,
kg2g = 1000,
g2kg = 0.001,
kJ2J = 1000,
J2kJ = 0.001,
se_median = 1.253,
frac2percent = 100
)

```

Arguments

cp	Specific heat of air for constant pressure (J K ⁻¹ kg ⁻¹)
Rgas	Universal gas constant (J mol ⁻¹ K ⁻¹)
Rv	Gas constant of water vapor (J kg ⁻¹ K ⁻¹) (Stull 1988 p.641)
Rd	Gas constant of dry air (J kg ⁻¹ K ⁻¹) (Foken p. 245)
Md	Molar mass of dry air (kg mol ⁻¹)
Mw	Molar mass of water vapor (kg mol ⁻¹)
eps	Ratio of the molecular weight of water vapor to dry air (=Mw/Md)
g	Gravitational acceleration (m s ⁻²)
solar_constant	Solar constant (W m ⁻²)
pressure0	Reference atmospheric pressure at sea level (Pa)
Tair0	Reference air temperature (K)
k	von Karman constant
Cmol	Molar mass of carbon (kg mol ⁻¹)
Omol	Molar mass of oxygen (kg mol ⁻¹)
H2Omol	Molar mass of water (kg mol ⁻¹)

sigma	Stefan-Boltzmann constant (W m ⁻² K ⁻⁴)
Pr	Prandtl number
Sc_CO2	Schmidt number for CO2
Le067	Lewis number for water vapor to the power of 0.67
Kelvin	Conversion degree Celsius to Kelvin
DwDc	Ratio of the molecular diffusivities for water vapor and CO2
days2seconds	Seconds per day
kPa2Pa	Conversion kilopascal (kPa) to pascal (Pa)
Pa2kPa	Conversion pascal (Pa) to kilopascal (kPa)
umol2mol	Conversion micromole (umol) to mole (mol)
mol2umol	Conversion mole (mol) to micromole (umol)
kg2g	Conversion kilogram (kg) to gram (g)
g2kg	Conversion gram (g) to kilogram (kg)
kJ2J	Conversion kilojoule (kJ) to joule (J)
J2kJ	Conversion joule (J) to kilojoule (kJ)
se_median	Conversion standard error (SE) of the mean to SE of the median
frac2percent	Conversion between fraction and percent

Details

This function is passed as an argument to every function that uses one or more constants. Individual constants passed to a function can be easily altered. E.g. the following command will change the value of the von Karman constant from 0.41 to 0.4:

```
bigleaf.constants(k=0.4)
```

the value of a constant can be returned by calling:

```
bigleaf.constants()$*name_of_constant*
```

To permanently change the constants contained within this function (which makes sense for some of them, e.g. for the von Karman constant), the command `fixInNamespace` can be used. E.g.

```
fixInNamespace(bigleaf.constants, ns="bigleaf")
```

Note that this has to be repeated every time the package is newly installed/loaded.

biochemical.energy *Biochemical Energy*

Description

Radiant energy absorbed in photosynthesis or heat release by respiration calculated from net ecosystem exchange of CO2 (NEE).

Usage

```
biochemical.energy(NEE, alpha = 0.422)
```

Arguments

NEE	Net ecosystem exchange (umol CO ₂ m ⁻² s ⁻¹)
alpha	Energy taken up/released by photosynthesis/respiration per mol CO ₂ fixed/respired (J umol ⁻¹)

Details

The following sign convention is employed: NEE is negative when carbon is taken up by the ecosystem. Positive values of the resulting biochemical energy mean that energy (heat) is taken up by the ecosystem, negative ones that heat is released. The value of alpha is taken from Nobel 1974 (see Meyers & Hollinger 2004), but other values have been used (e.g. Blanken et al., 1997)

Value

Sp - biochemical energy (W m⁻²)

References

Meyers, T.P., Hollinger, S.E. 2004: An assessment of storage terms in the surface energy balance of maize and soybean. *Agricultural and Forest Meteorology* 125, 105-115.

Nobel, P.S., 1974: *Introduction to Biophysical Plant Physiology*. Freeman, New York.

Blanken, P.D. et al., 1997: Energy balance and canopy conductance of a boreal aspen forest: Partitioning overstory and understory components. *Journal of Geophysical Research* 102, 28915-28927.

Examples

```
# Calculate biochemical energy taken up by the ecosystem with
# a measured NEE of -30umol CO2 m-2 s-1
biochemical.energy(NEE=-30)
```

decoupling

Canopy-Atmosphere Decoupling Coefficient

Description

The canopy-atmosphere decoupling coefficient 'Omega'.

Usage

```

decoupling(
  data,
  Tair = "Tair",
  pressure = "pressure",
  Ga = "Ga_h",
  Gs = "Gs_ms",
  approach = c("Jarvis&McNaughton_1986", "Martin_1989"),
  LAI,
  Esat.formula = c("Sonntag_1990", "Alduchov_1996", "Allen_1998"),
  constants = bigleaf.constants()
)

```

Arguments

data	Data.frame or matrix containing all required input variables
Tair	Air temperature (deg C)
pressure	Atmospheric pressure (kPa)
Ga	Aerodynamic conductance to heat/water vapor (m s-1)
Gs	Surface conductance (m s-1)
approach	Approach used to calculate omega. Either "Jarvis&McNaughton_1986" (default) or "Martin_1989".
LAI	Leaf area index (m ² m ⁻²), only used if approach = "Martin_1989".
Esat.formula	Optional: formula to be used for the calculation of esat and the slope of esat. One of "Sonntag_1990" (Default), "Alduchov_1996", or "Allen_1998". See Esat.slope .
constants	Kelvin - conversion degree Celsius to Kelvin cp - specific heat of air for constant pressure (J K ⁻¹ kg ⁻¹) eps - ratio of the molecular weight of water vapor to dry air (-) sigma - Stefan-Boltzmann constant (W m ⁻² K ⁻⁴) Pa2kPa - conversion pascal (Pa) to kilopascal (kPa)

Details

The decoupling coefficient Omega ranges from 0 to 1 and quantifies the linkage of the conditions (foremost humidity and temperature) at the canopy surface to the ambient air. Values close to 0 indicate well coupled conditions characterized by high physiological (i.e. stomatal) control on transpiration and similar conditions at the canopy surface compared to the atmosphere above the canopy. Values close to 1 indicate the opposite, i.e. decoupled conditions and a low stomatal control on transpiration (Jarvis & McNaughton 1986).

The "Jarvis&McNaughton_1986" approach (default option) is the original formulation for the decoupling coefficient, given by (for an amphistomatous canopy):

$$\Omega = \frac{\epsilon + 1}{\epsilon + 1 + \frac{G_a}{G_c}}$$

where $\epsilon = \frac{s}{\gamma}$ is a dimensionless coefficient with s being the slope of the saturation vapor pressure curve (Pa K⁻¹), and γ the psychrometric constant (Pa K⁻¹).

The approach "Martin_1989" by Martin 1989 additionally takes radiative coupling into account:

$$\Omega = \frac{\epsilon + 1 + \frac{Gr}{Ga}}{\epsilon + (1 + \frac{Ga}{Gs})(1 + \frac{Gr}{Ga})}$$

Value

Ω - the decoupling coefficient Omega (-)

References

Jarvis P.G., McNaughton K.G., 1986: Stomatal control of transpiration: scaling up from leaf to region. *Advances in Ecological Research* 15, 1-49.

Martin P., 1989: The significance of radiative coupling between vegetation and the atmosphere. *Agricultural and Forest Meteorology* 49, 45-53.

See Also

[aerodynamic.conductance](#), [surface.conductance](#), [equilibrium.imposed.ET](#)

Examples

```
# Omega calculated following Jarvis & McNaughton 1986
set.seed(3)
df <- data.frame(Tair=rnorm(20,25,1),pressure=100,Ga_h=rnorm(20,0.06,0.01),
                 Gs_ms=rnorm(20,0.005,0.001))
decoupling(df,approach="Jarvis&McNaughton_1986")

# Omega calculated following Martin 1989 (requires LAI)
decoupling(df,approach="Martin_1989",LAI=4)
```

dew.point

Dew Point

Description

calculates the dew point, the temperature to which air must be cooled to become saturated (i.e. $e = \text{Esat}(T_d)$)

Usage

```
dew.point(  
  Tair,  
  VPD,  
  accuracy = 0.001,  
  Esat.formula = c("Sonntag_1990", "Alduchov_1996", "Allen_1998"),  
  constants = bigleaf.constants()  
)
```

Arguments

Tair	Air temperature (degC)
VPD	Vapor pressure deficit (kPa)
accuracy	Accuracy of the result (deg C)
Esat.formula	Optional: formula to be used for the calculation of esat and the slope of esat. One of "Sonntag_1990" (Default), "Alduchov_1996", or "Allen_1998". See Esat.slope .
constants	Pa2kPa - conversion pascal (Pa) to kilopascal (kPa)

Details

Dew point temperature (Td) is defined by:

$$e = Esat(Td)$$

where e is vapor pressure of the air and Esat is the vapor pressure deficit. This equation is solved for Td using [optimize](#).

Value

Td - dew point temperature (degC)

References

Monteith J.L., Unsworth M.H., 2008: Principles of Environmental Physics. 3rd edition. Academic Press, London.

Examples

```
dew.point(c(25,30),1.5)
```

DE_Tha_Jun_2014

*Eddy Covariance Data of DE-Tha (Tharandt)***Description**

Halfhourly eddy covariance Data of the site DE-Tha, a spruce forest in Eastern Germany (<https://sites.fluxdata.org/DE-Tha/>). Data are from June 2014.

Usage

DE_Tha_Jun_2014

Format

A data frame with 1440 observations and 32 columns:

year year of measurement

month month of measurement

doy day of year

hour hour (0 - 23.5)

Tair Air temperature (degC) [TA_F]

Tair_qc Quality control of Tair [TA_F_QC]

PPFD Photosynthetic photon flux density (umol m⁻² s⁻¹) [PPFD_IN]

PPFD_qc Quality control of PPFD [PPFD_IN_QC]

VPD Vapor pressure deficit (kPa) [VPD_F]

VPD_qc Quality control of VPD [VPD_F_QC]

pressure Atmospheric pressure (kPa) [PA_F]

precip precipitation (mm) [P_F]

precip_qc Quality control of precip [P_F_QC]

ustar friction velocity (m s⁻¹) [USTAR]

wind horizontal wind velocity (m s⁻¹) [WS_F]

wind_qc Quality control of wind [WS_F_QC]

Ca CO₂ concentration (ppm) [CO₂_F_MDS]

Ca_qc Quality control of Ca [CO₂_F_MDS_QC]

LW_up upward longwave radiation (W m⁻²) [LW_OUT]

LW_down downward longwave radiation (W m⁻²) [LW_IN_F]

Rn Net radiation (W m⁻²) [NETRAD]

LE Latent heat flux (W m⁻²) [LE_F_MDS]

LE_qc Quality control of LE [LE_F_MDS_QC]

H Sensible heat flux (W m⁻²) [H_F_MDS]

- H_qc** Quality control of H [H_F_MDS_QC]
G Ground heat flux (W m⁻²) [G_F_MDS]
G_qc Quality control of G [G_F_MDS_QC]
NEE Net ecosystem exchange (umol m⁻² s⁻¹) [NEE_VUT_USTAR50]
NEE_qc Quality control of NEE [NEE_VUT_USTAR50_QC]
GPP Gross primary productivity from nighttime partitioning (umol m⁻² s⁻¹) [GPP_NT_VUT_USTAR50]
GPP_qc Quality control of GPP [NEE_VUT_USTAR50_QC]
Reco Ecosystem respiration from nighttime partitioning (umol m⁻² s⁻¹) [RECO_NT_VUT_USTAR50]

Note

The original variable names as provided by the FLUXNET2015 dataset are given in squared brackets. Note that variable units have been converted in some cases (e.g. VPD from hPa to kPa).

Source

original data were downloaded from <https://fluxnet.org/> (accessed 09 November 2016)

energy.closure	<i>Energy Balance Closure</i>
----------------	-------------------------------

Description

Calculates the degree of the energy balance non-closure for the entire time span based on the ratio of two sums (energy balance ratio), and ordinary least squares (OLS).

Usage

```
energy.closure(  
  data,  
  Rn = "Rn",  
  G = NULL,  
  S = NULL,  
  LE = "LE",  
  H = "H",  
  instantaneous = FALSE,  
  missing.G.as.NA = FALSE,  
  missing.S.as.NA = FALSE  
)
```

Arguments

data	Data.frame or matrix containing all required variables.
Rn	Net radiation (W m-2)
G	Ground heat flux (W m-2); optional
S	Sum of all storage fluxes (W m-2); optional
LE	Latent heat flux (W m-2)
H	Sensible heat flux (W m-2)
instantaneous	should the energy balance be calculated at the time step of the observations (TRUE), or over the entire time period provided as input (FALSE)
missing.G.as.NA	if TRUE, missing G are treated as NAs ,otherwise set to 0.
missing.S.as.NA	if TRUE, missing S are treated as NAs, otherwise set to 0.

Details

The energy balance ratio (EBR) is calculated as:

$$EBR = \text{sum}(LE + H) / \text{sum}(Rn - G - S)$$

the sum is taken for all time steps with complete observations (i.e. where all energy balance terms are available).

Value

a named vector containing:

n	number of complete (all energy balance terms available) observations
intercept	intercept of the OLS regression
slope	slope of the OLS regression
r_squared	r ² of the OLS regression
EBR	energy balance ratio

if instantaneous = TRUE, only EBR is returned.

References

Wilson K., et al. 2002: Energy balance closure at FLUXNET sites. *Agricultural and Forest Meteorology* 113, 223-243.

Examples

```
## characterize energy balance closure for DE-Tha in June 2014
energy.closure(DE_Tha_Jun_2014, instantaneous=FALSE)

## look at half-hourly closure
EBR_inst <- energy.closure(DE_Tha_Jun_2014, instantaneous=TRUE)
summary(EBR_inst)
```

energy.use.encyclopedia *Energy-Use Efficiency (EUE)*

Description

Fraction of net radiation fixed by primary productivity.

Usage

```
energy.use.encyclopedia(GPP, alpha = 0.422, Rn)
```

Arguments

GPP	Gross primary productivity exchange (umol CO ₂ m ⁻² s ⁻¹)
alpha	Energy taken up/released by photosynthesis/respiration (J umol ⁻¹)
Rn	Net radiation (W m ⁻²)

Details

Energy use efficiency is calculated as:

$$EUE = \text{sum}(GPP) / \text{sum}(Rn)$$

where the sums are calculated for complete cases of GPP and Rn over the entire time period.

Value

EUE - Energy use efficiency (-)

See Also

[light.use.encyclopedia](#)

Examples

```
energy.use.encyclopedia(GPP=20,Rn=500)
```

equilibrium.imposed.ET

Equilibrium and Imposed Evapotranspiration

Description

Evapotranspiration (ET) split up into imposed ET and equilibrium ET.

Usage

```
equilibrium.imposed.ET(
  data,
  Tair = "Tair",
  pressure = "pressure",
  VPD = "VPD",
  Gs = "Gs_ms",
  Rn = "Rn",
  G = NULL,
  S = NULL,
  missing.G.as.NA = FALSE,
  missing.S.as.NA = FALSE,
  Esat.formula = c("Sonntag_1990", "Alduchov_1996", "Allen_1998"),
  constants = bigleaf.constants()
)
```

Arguments

data	Data.frame or matrix containing all required input variables
Tair	Air temperature (deg C)
pressure	Atmospheric pressure (kPa)
VPD	Air vapor pressure deficit (kPa)
Gs	surface conductance to water vapor (m s-1)
Rn	Net radiation (W m-2)
G	Ground heat flux (W m-2); optional
S	Sum of all storage fluxes (W m-2); optional
missing.G.as.NA	if TRUE, missing G are treated as NAs, otherwise set to 0.
missing.S.as.NA	if TRUE, missing S are treated as NAs, otherwise set to 0.
Esat.formula	Optional: formula to be used for the calculation of esat and the slope of esat. One of "Sonntag_1990" (Default), "Alduchov_1996", or "Allen_1998". See Esat.slope .
constants	cp - specific heat of air for constant pressure (J K-1 kg-1) eps - ratio of the molecular weight of water vapor to dry air (-) Pa2kPa - conversion pascal (Pa) to kilopascal (kPa)

Details

Total evapotranspiration can be written in the form (Jarvis & McNaughton 1986):

$$ET = \Omega ET_{eq} + (1 - \Omega) ET_{imp}$$

where Ω is the decoupling coefficient as calculated from [decoupling](#). ET_{eq} is the equilibrium evapotranspiration rate, the ET rate that would occur under uncoupled conditions, where the heat budget is dominated by radiation (when $G_a \rightarrow 0$):

$$ET_{eq} = (\Delta * (R_n - G - S) * \lambda) / (\Delta + \gamma)$$

where Δ is the slope of the saturation vapor pressure curve (kPa K⁻¹), λ is the latent heat of vaporization (J kg⁻¹), and γ is the psychrometric constant (kPa K⁻¹). ET_{imp} is the imposed evapotranspiration rate, the ET rate that would occur under fully coupled conditions (when $G_a \rightarrow \infty$):

$$ET_{imp} = (\rho * c_p * VPD * G_s * \lambda) / \gamma$$

where ρ is the air density (kg m⁻³).

Value

A data.frame with the following columns:

ET_eq	Equilibrium ET (kg m ⁻² s ⁻¹)
ET_imp	Imposed ET (kg m ⁻² s ⁻¹)
LE_eq	Equilibrium LE (W m ⁻²)
LE_imp	Imposed LE (W m ⁻²)

Note

Surface conductance (G_s) can be calculated with [surface.conductance](#). Aerodynamic conductance (G_a) can be calculated using [aerodynamic.conductance](#).

References

Jarvis, P.G., McNaughton, K.G., 1986: Stomatal control of transpiration: scaling up from leaf to region. *Advances in Ecological Research* 15, 1-49.

Monteith, J.L., Unsworth, M.H., 2008: *Principles of Environmental Physics*. 3rd edition. Academic Press, London.

See Also

[decoupling](#)

Examples

```
df <- data.frame(Tair=20,pressure=100,VPD=seq(0.5,4,0.5),
                 Gs_ms=seq(0.01,0.002,length.out=8),Rn=seq(50,400,50))
equilibrium.imposed.ET(df)
```

 Esat.slope

Saturation Vapor Pressure (Esat) and Slope of the Esat Curve

Description

Calculates saturation vapor pressure (Esat) over water and the corresponding slope of the saturation vapor pressure curve.

Usage

```
Esat.slope(
  Tair,
  formula = c("Sonntag_1990", "Alduchov_1996", "Allen_1998"),
  constants = bigleaf.constants()
)
```

Arguments

Tair	Air temperature (deg C)
formula	Formula to be used. Either "Sonntag_1990" (Default), "Alduchov_1996", or "Allen_1998".
constants	Pa2kPa - conversion pascal (Pa) to kilopascal (kPa)

Details

Esat (kPa) is calculated using the Magnus equation:

$$Esat = a * exp((b * Tair)/(c + Tair))/1000$$

where the coefficients a, b, c take different values depending on the formula used. The default values are from Sonntag 1990 (a=611.2, b=17.62, c=243.12). This version of the Magnus equation is recommended by the WMO (WMO 2008; p1.4-29). Alternatively, parameter values determined by Alduchov & Eskridge 1996 or Allen et al. 1998 can be used (see references). The slope of the Esat curve (Δ) is calculated as the first derivative of the function:

$$\Delta = dEsat/dTair$$

which is solved using [D](#).

Value

A dataframe with the following columns:

Esat	Saturation vapor pressure (kPa)
Delta	Slope of the saturation vapor pressure curve (kPa K-1)

References

- Sonntag D. 1990: Important new values of the physical constants of 1986, vapor pressure formulations based on the ITS-90 and psychrometric formulae. *Zeitschrift fuer Meteorologie* 70, 340-344.
- World Meteorological Organization 2008: *Guide to Meteorological Instruments and Methods of Observation (WMO-No.8)*. World Meteorological Organization, Geneva. 7th Edition.
- Alduchov, O. A. & Eskridge, R. E., 1996: Improved Magnus form approximation of saturation vapor pressure. *Journal of Applied Meteorology*, 35, 601-609
- Allen, R.G., Pereira, L.S., Raes, D., Smith, M., 1998: *Crop evapotranspiration - Guidelines for computing crop water requirements - FAO irrigation and drainage paper 56*, FAO, Rome.

Examples

```
Esat.slope(seq(0,45,5))[, "Esat"] # Esat in kPa
Esat.slope(seq(0,45,5))[, "Delta"] # the corresponding slope of the Esat curve (Delta) in kPa K-1
```

```
extraterrestrial.radiation
      Extraterrestrial solar radiation
```

Description

Compute the extraterrestrial solar radiation with the

Usage

```
extraterrestrial.radiation(doy, constants = bigleaf.constants())
```

Arguments

doy	integer vector with day of year (DoY)
constants	solar_constant - solar constant (W m-2)

Details

Computation follows Lanini, 2010 (Master thesis, Bern University)

Value

numeric vector of extraterrestrial radiation (W_m-2)

Examples

```
plot(1:365, extraterrestrial.radiation(1:365), type = "l"
     , ylab = "radiation (W m-2)", xlab = "day of year")
```

Description

Filters time series of EC data for high-quality values and specified meteorological conditions.

Usage

```
filter.data(  
  data,  
  quality.control = TRUE,  
  filter.growseas = FALSE,  
  filter.precip = FALSE,  
  filter.vars = NULL,  
  filter.vals.min,  
  filter.vals.max,  
  NA.as.invalid = TRUE,  
  vars.qc = NULL,  
  quality.ext = "_qc",  
  good.quality = c(0, 1),  
  missing.qc.as.bad = TRUE,  
  GPP = "GPP",  
  doy = "doy",  
  year = "year",  
  tGPP = 0.5,  
  ws = 15,  
  min.int = 5,  
  precip = "precip",  
  tprecip = 0.01,  
  precip.hours = 24,  
  records.per.hour = 2,  
  filtered.data.to.NA = TRUE,  
  constants = bigleaf.constants()  
)
```

Arguments

data	Data.frame or matrix containing all required input variables in half-hourly or hourly resolution. Including year, month, day information
quality.control	Should quality control be applied? Defaults to TRUE.
filter.growseas	Should data be filtered for growing season? Defaults to FALSE.
filter.precip	Should precipitation filtering be applied? Defaults to FALSE.
filter.vars	Additional variables to be filtered. Vector of type character.

<code>filter.vals.min</code>	Minimum values of the variables to be filtered. Numeric vector of the same length than <code>filter.vars</code> . Set to NA to be ignored.
<code>filter.vals.max</code>	Maximum values of the variables to be filtered. Numeric vector of the same length than <code>filter.vars</code> . Set to NA to be ignored.
<code>NA.as.invalid</code>	If TRUE (the default) missing data are filtered out (applies to all variables).
<code>vars.qc</code>	Character vector indicating the variables for which quality filter should be applied. Ignored if <code>quality.control = FALSE</code> .
<code>quality.ext</code>	The extension to the variables' names that marks them as quality control variables. Ignored if <code>quality.control = FALSE</code> .
<code>good.quality</code>	Which values indicate good quality (i.e. not to be filtered) in the quality control (qc) variables? Ignored if <code>quality.control = FALSE</code> .
<code>missing.qc.as.bad</code>	If quality control variable is NA, should the corresponding data point be treated as bad quality? Defaults to TRUE. Ignored if <code>quality.control = FALSE</code> .
<code>GPP</code>	Gross primary productivity (umol m ⁻² s ⁻¹); Ignored if <code>filter.growseas = FALSE</code> .
<code>doy</code>	Day of year; Ignored if <code>filter.growseas = FALSE</code> .
<code>year</code>	Year; Ignored if <code>filter.growseas = FALSE</code> .
<code>tGPP</code>	GPP threshold (fraction of 95th percentile of the GPP time series). Must be between 0 and 1. Ignored if <code>filter.growseas</code> is FALSE.
<code>ws</code>	Window size used for GPP time series smoothing. Ignored if <code>filter.growseas = FALSE</code> .
<code>min.int</code>	Minimum time interval in days for a given state of growing season. Ignored if <code>filter.growseas = FALSE</code> .
<code>precip</code>	Precipitation (mm time ⁻¹)
<code>tprecip</code>	Precipitation threshold used to identify a precipitation event (mm). Ignored if <code>filter.precip = FALSE</code> .
<code>precip.hours</code>	Number of hours removed following a precipitation event (h). Ignored if <code>filter.precip = FALSE</code> .
<code>records.per.hour</code>	Number of observations per hour. I.e. 2 for half-hourly data.
<code>filtered.data.to.NA</code>	Logical. If TRUE (the default), all variables in the input <code>data.frame/matrix</code> are set to NA for the time step where ANY of the <code>filter.vars</code> were beyond their acceptable range (as determined by <code>filter.vals.min</code> and <code>filter.vals.max</code>). If FALSE, values are not filtered, and an additional column 'valid' is added to the <code>data.frame/matrix</code> , indicating if any value of a row did (1) or did not fulfill the filter criteria (0).
<code>constants</code>	<code>frac2percent</code> - conversion between fraction and percent

Details

This routine consists of two parts:

1) Quality control: All variables included in `vars.qc` are filtered for good quality data. For these variables a corresponding quality variable with the same name as the variable plus the extension as specified in `quality.ext` must be provided. For time steps where the value of the quality indicator is not included in the argument `good.quality`, i.e. the quality is not considered as 'good', its value is set to NA.

2) Meteorological filtering. Under certain conditions (e.g. low `ustar`), the assumptions of the EC method are not fulfilled. Further, some data analysis require certain meteorological conditions, such as periods without rainfall, or active vegetation (growing season, daytime). The filter applied in this second step serves to exclude time periods that do not fulfill the criteria specified in the arguments. More specifically, time periods where one of the variables is higher or lower than the specified thresholds (`filter.vals.min` and `filter.vals.max`) are set to NA for all variables. If a threshold is set to NA, it will be ignored.

Value

If `filtered.data.to.NA = TRUE` (default), the input `data.frame/matrix` with observations which did not fulfill the filter criteria set to NA. If `filtered.data.to.NA = FALSE`, the input `data.frame/matrix` with an additional column "valid", which indicates whether all the data of a time step fulfill the filtering criteria (1) or not (0).

Note

The thresholds set with `filter.vals.min` and `filter.vals.max` filter all data that are smaller than (" $<$ "), or greater than (" $>$ ") the specified thresholds. That means if a variable has exactly the same value as the threshold, it will not be filtered. Likewise, `tprecip` filters all data that are greater than `tprecip`.

Variables considered of bad quality (as specified by the corresponding quality control variables) will be set to NA by this routine. Data that do not fulfill the filtering criteria are set to NA if `filtered.data.to.NA = TRUE`. Note that with this option `*all*` variables of the same time step are set to NA. Alternatively, if `filtered.data.to.NA = FALSE` data are not set to NA, and a new column "valid" is added to the `data.frame/matrix`, indicating if any value of a row did (1) or did not fulfill the filter criteria (0).

Examples

```
# Example of data filtering; data are for a month within the growing season,
# hence growing season is not filtered.
# If filtered.data.to.NA=TRUE, all values of a row are set to NA if one filter
# variable is beyond its bounds.
DE_Tha_Jun_2014_2 <- filter.data(DE_Tha_Jun_2014,quality.control=FALSE,
                                vars.qc=c("Tair","precip","H","LE"),
                                filter.growseas=FALSE,filter.precip=TRUE,
                                filter.vars=c("Tair","PPFD","ustar"),
                                filter.vals.min=c(5,200,0.2),
                                filter.vals.max=c(NA,NA,NA),NA.as.invalid=TRUE,
                                quality.ext="_qc",good.quality=c(0,1),
                                missing.qc.as.bad=TRUE,GPP="GPP",doy="doy",
```

```

year="year", tGPP=0.5, ws=15, min.int=5, precip="precip",
tprecip=0.1, precip.hours=24, records.per.hour=2,
filtered.data.to.NA=TRUE)

## same, but with filtered.data.to.NA=FALSE
DE_Tha_Jun_2014_3 <- filter.data(DE_Tha_Jun_2014, quality.control=FALSE,
vars.qc=c("Tair", "precip", "H", "LE"),
filter.growseas=FALSE, filter.precip=TRUE,
filter.vars=c("Tair", "PPFD", "ustar"),
filter.vals.min=c(5, 200, 0.2),
filter.vals.max=c(NA, NA, NA), NA.as.invalid=TRUE,
quality.ext="_qc", good.quality=c(0, 1),
missing.qc.as.bad=TRUE, GPP="GPP", doy="doy",
year="year", tGPP=0.5, ws=15, min.int=5, precip="precip",
tprecip=0.1, precip.hours=24, records.per.hour=2,
filtered.data.to.NA=FALSE)

# note the additional column 'valid' in DE_Tha_Jun_2014_3.
# To remove time steps marked as filtered out (i.e. 0 values in column 'valid'):
DE_Tha_Jun_2014_3[DE_Tha_Jun_2014_3["valid"] == 0,] <- NA

```

filter.growing.season *GPP-based Growing Season Filter*

Description

Filters annual time series for growing season based on smoothed daily GPP data.

Usage

```
filter.growing.season(GPPd, tGPP, ws = 15, min.int = 5)
```

Arguments

GPPd	daily GPP (any unit)
tGPP	GPP threshold (fraction of 95th percentile of the GPP time series). Takes values between 0 and 1.
ws	window size used for GPP time series smoothing
min.int	minimum time interval in days for a given state of growing season

Details

The basic idea behind the growing season filter is that vegetation is considered to be active when its carbon uptake (GPP) is above a specified threshold, which is defined relative to the peak GPP (95th percentile) observed in the year. The GPP-threshold is calculated as:

$$GPP_{threshold} = \text{quantile}(GPPd, 0.95) * tGPP$$

GPPd time series are smoothed with a moving average to avoid fluctuations in the delineation of the growing season. The window size defaults to 15 days, but depending on the ecosystem, other values can be appropriate.

The argument `min.int` serves to avoid short fluctuations in the status growing season vs. no growing season by defining a minimum length of the status. If a time interval shorter than `min.int` is labeled as growing season or non-growing season, it is changed to the status of the neighboring values.

Value

a vector of type integer of the same length as the input GPPd in which 0 indicate no growing season (dormant season) and 1 indicate growing season.

FR_Pue_May_2012

Eddy Covariance Data of FR-Pue (Puechabon)

Description

Halfhourly eddy covariance Data of the site FR-Pue, a Mediterranean evergreen oak forest in Southern France (<https://sites.fluxdata.org/FR-Pue/>). Data are from May 2012.

Usage

FR_Pue_May_2012

Format

A data frame with 1488 observations and 29 columns:

year year of measurement

month month of measurement

doy day of year

hour hour (0 - 23.5)

Tair Air temperature (degC) [TA_F]

Tair_qc Quality control of Tair [TA_F_QC]

PPFD Photosynthetic photon flux density (umol m⁻² s⁻¹) [PPFD_IN]

PPFD_qc Quality control of PPFD [PPFD_IN_QC]

VPD Vapor pressure deficit (kPa) [VPD_F]

VPD_qc Quality control of VPD [VPD_F_QC]

pressure Atmospheric pressure (kPa) [PA_F]

precip precipitation (mm) [P_F]
precip_qc Quality control of precip [P_F_QC]
ustar friction velocity (m s-1) [USTAR]
wind horizontal wind velocity (m s-1) [WS_F]
wind_qc Quality control of wind [WS_F_QC]
Ca CO2 concentration (ppm) [CO2_F_MDS]
Ca_qc Quality control of Ca [CO2_F_MDS_QC]
LW_up upward longwave radiation (W m-2) [LW_OUT]
Rn Net radiation (W m-2) [NETRAD]
LE Latent heat flux (W m-2) [LE_F_MDS]
LE_qc Quality control of LE [LE_F_MDS_QC]
H Sensible heat flux (W m-2) [H_F_MDS]
H_qc Quality control of H [H_F_MDS_QC]
NEE Net ecosystem exchange (umol m-2 s-1) [NEE_VUT_USTAR50]
NEE_qc Quality control of NEE [NEE_VUT_USTAR50_QC]
GPP Gross primary productivity from nighttime partitioning (umol m-2 s-1) [GPP_NT_VUT_USTAR50]
GPP_qc Quality control of GPP [NEE_VUT_USTAR50_QC]
Reco Ecosystem respiration from nighttime partitioning (umol m-2 s-1) [RECO_NT_VUT_USTAR50]

Note

The original variable names as provided by the FLUXNET2015 dataset are given in squared brackets. Note that variable units have been converted in some cases (e.g. VPD from hPa to kPa).

Source

original data were downloaded from <https://fluxnet.org/> (accessed 09 November 2016)

Gb.Choudhury

Boundary Layer Conductance according to Choudhury & Monteith 1988

Description

A formulation for the canopy boundary layer conductance for heat transfer according to Choudhury & Monteith 1988.

Usage

```

Gb.Choudhury(
  data,
  Tair = "Tair",
  pressure = "pressure",
  wind = "wind",
  ustar = "ustar",
  H = "H",
  leafwidth,
  LAI,
  zh,
  zr,
  d,
  z0m = NULL,
  stab_formulation = c("Dyer_1970", "Businger_1971"),
  Sc = NULL,
  Sc_name = NULL,
  constants = bigleaf.constants()
)

```

Arguments

data	Data.frame or matrix containing all required variables
Tair	Air temperature (degC)
pressure	Atmospheric pressure (kPa)
wind	Wind speed at sensor height (m s-1)
ustar	Friction velocity (m s-1)
H	Sensible heat flux (W m-2)
leafwidth	Leaf width (m)
LAI	One-sided leaf area index
zh	Canopy height (m)
zr	Instrument (reference) height (m)
d	Zero-plane displacement height (-), can be calculated using roughness.parameters
z0m	Roughness length for momentum (m). If not provided, calculated from roughness.parameters within wind.profile
stab_formulation	Stability correction function used (If stab_correction = TRUE). Either "Dyer_1970" or "Businger_1971".
Sc	Optional: Schmidt number of additional quantities to be calculated
Sc_name	Optional: Name of the additional quantities, has to be of same length than Sc_name
constants	k - von-Karman constant Sc_CO2 - Schmidt number for CO2 Pr - Prandtl number (if Sc is provided)

Details

Boundary layer conductance according to Choudhury & Monteith 1988 is given by:

$$Gb_h = LAI((2a/\alpha) * \text{sqrt}(u(h)/w) * (1 - \exp(-\alpha/2)))$$

where $u(zh)$ is the wind speed at the canopy surface, approximated from measured wind speed at sensor height zr and a wind extinction coefficient α :

$$u(zh) = u(zr)/(\exp(\alpha(zr/zh - 1)))$$

α is modeled as an empirical relation to LAI (McNaughton & van den Hurk 1995):

$$\alpha = 4.39 - 3.97 * \exp(-0.258 * LAI)$$

$G_b (=1/R_b)$ for water vapor and heat are assumed to be equal in this package. G_b for other quantities x is calculated as (Hicks et al. 1987):

$$Gb_x = G_b / (Sc_x / Pr)^{0.67}$$

where Sc_x is the Schmidt number of quantity x , and Pr is the Prandtl number (0.71).

Value

A data frame with the following columns:

Gb_h	Boundary layer conductance for heat transfer (m s-1)
Rb_h	Boundary layer resistance for heat transfer (s m-1)
kB_h	kB-1 parameter for heat transfer
Gb_Sc_name	Boundary layer conductance for Sc_name (m s-1). Only added if Sc_name and Sc_name are provided

Note

If the roughness length for momentum ($z0m$) is not provided as input, it is estimated from the function `roughness.parameters` within `wind.profile`. This function estimates a single $z0m$ value for the entire time period! If a varying $z0m$ value (e.g. across seasons or years) is required, $z0m$ should be provided as input argument.

References

- Choudhury, B. J., Monteith J.L., 1988: A four-layer model for the heat budget of homogeneous land surfaces. *Q. J. R. Meteorol. Soc.* 114, 373-398.
- McNaughton, K. G., Van den Hurk, B.J.J.M., 1995: A 'Lagrangian' revision of the resistors in the two-layer model for calculating the energy budget of a plant canopy. *Boundary-Layer Meteorology* 74, 261-288.
- Hicks, B.B., Baldocchi, D.D., Meyers, T.P., Hosker, J.R., Matt, D.R., 1987: A preliminary multiple resistance routine for deriving dry deposition velocities from measured quantities. *Water, Air, and Soil Pollution* 36, 311-330.

See Also

[Gb.Thom](#), [Gb.Su](#), [aerodynamic.conductance](#)

Examples

```
## bulk canopy boundary layer resistance for a closed canopy (LAI=5)
## with large leaves (leafwidth=0.1)
df <- data.frame(Tair=25,pressure=100,wind=c(3,4,5),ustar=c(0.5,0.6,0.65),H=c(200,230,250))
Gb.Choudhury(data=df,leafwidth=0.1,LAI=5,zh=25,d=17.5,zr=40)

## same conditions, but smaller leaves (leafwidth=0.01)
Gb.Choudhury(data=df,leafwidth=0.01,LAI=5,zh=25,d=17.5,zr=40)
```

Gb.Su

Boundary Layer Conductance according to Su et al. 2001

Description

A physically based formulation for the canopy boundary layer conductance to heat transfer according to Su et al. 2001.

Usage

```
Gb.Su(
  data,
  Tair = "Tair",
  pressure = "pressure",
  ustar = "ustar",
  wind = "wind",
  H = "H",
  zh,
  zr,
  d,
  z0m = NULL,
  D1,
  fc = NULL,
  LAI = NULL,
  N = 2,
  Cd = 0.2,
  hs = 0.01,
  stab_formulation = c("Dyer_1970", "Businger_1971"),
  Sc = NULL,
  Sc_name = NULL,
  constants = bigleaf.constants()
)
```

Arguments

<code>data</code>	Data.frame or matrix containing all required variables
<code>Tair</code>	Air temperature (degC)
<code>pressure</code>	Atmospheric pressure (kPa)
<code>ustar</code>	Friction velocity (m s-1)
<code>wind</code>	Wind speed (m s-1)
<code>H</code>	Sensible heat flux (W m-2)
<code>zh</code>	Canopy height (m)
<code>zr</code>	Reference height (m)
<code>d</code>	Zero-plane displacement height (-), can be calculated using <code>roughness.parameters</code>
<code>z0m</code>	Roughness length for momentum (m). If not provided, calculated from <code>roughness.parameters</code> within <code>wind.profile</code>
<code>Dl</code>	Leaf characteristic dimension (m)
<code>fc</code>	Fractional vegetation cover [0-1] (if not provided, calculated from LAI)
<code>LAI</code>	One-sided leaf area index (-)
<code>N</code>	Number of leaf sides participating in heat exchange (defaults to 2)
<code>Cd</code>	Foliage drag coefficient (-)
<code>hs</code>	Roughness height of the soil (m)
<code>stab_formulation</code>	Stability correction function used (If <code>stab_correction = TRUE</code>). Either "Dyer_1970" or "Businger_1971".
<code>Sc</code>	Optional: Schmidt number of additional quantities to be calculated
<code>Sc_name</code>	Optional: Name of the additional quantities, has to be of same length than <code>Sc_name</code>
<code>constants</code>	Kelvin - conversion degree Celsius to Kelvin <code>pressure0</code> - reference atmospheric pressure at sea level (Pa) <code>Tair0</code> - reference air temperature (K) <code>Sc_CO2</code> - Schmidt number for CO2 <code>Pr</code> - Prandtl number (if <code>Sc</code> is provided)

Details

The formulation is based on the kB-1 model developed by Massman 1999. Su et al. 2001 derived the following approximation:

$$kB - 1 = (kCdfc^2)/(4Ctustar/u(zh)) + kB_s - 1(1 - fc)^2$$

If `fc` (fractional vegetation cover) is missing, it is estimated from LAI:

$$fc = 1 - \exp(-LAI/2)$$

The wind speed at the top of the canopy is calculated using function [wind.profile](#).

Ct is the heat transfer coefficient of the leaf (Massman 1999):

$$Ct = Pr^{-2/3} Reh^{-1/2} N$$

where Pr is the Prandtl number (set to 0.71), and Reh is the Reynolds number for leaves:

$$Reh = Dwind(zh)/v$$

kBs-1, the kB-1 value for bare soil surface, is calculated according to Su et al. 2001:

$$kB_s^{-1} = 2.46(Re)^{0.25} - \ln(7.4)$$

Gb (=1/Rb) for water vapor and heat are assumed to be equal in this package. Gb for other quantities x is calculated as (Hicks et al. 1987):

$$Gb_x = Gb / (Sc_x / Pr)^{0.67}$$

where Sc_x is the Schmidt number of quantity x, and Pr is the Prandtl number (0.71).

Value

A data.frame with the following columns:

Gb_h	Boundary layer conductance for heat transfer (m s-1)
Rb_h	Boundary layer resistance for heat transfer (s m-1)
kB_h	kB-1 parameter for heat transfer
Gb_Sc_name	Boundary layer conductance for Sc_name (m s-1). Only added if Sc_name and Sc_name are provided

Note

If the roughness length for momentum (z0m) is not provided as input, it is estimated from the function roughness.parameters within wind.profile. This function estimates a single z0m value for the entire time period! If a varying z0m value (e.g. across seasons or years) is required, z0m should be provided as input argument.

References

- Su, Z., Schmugge, T., Kustas, W. & Massman, W., 2001: An evaluation of two models for estimation of the roughness height for heat transfer between the land surface and the atmosphere. *Journal of Applied Meteorology* 40, 1933-1951.
- Massman, W., 1999: A model study of kB H- 1 for vegetated surfaces using 'localized near-field' Lagrangian theory. *Journal of Hydrology* 223, 27-43.
- Hicks, B.B., Baldocchi, D.D., Meyers, T.P., Hosker, J.R., Matt, D.R., 1987: A preliminary multiple resistance routine for deriving dry deposition velocities from measured quantities. *Water, Air, and Soil Pollution* 36, 311-330.

See Also

[Gb.Thom](#), [Gb.Choudhury](#), [aerodynamic.conductance](#)

Examples

```
# Canopy boundary layer resistance (and kB-1 parameter) for a set of meteorological conditions,
# a leaf characteristic dimension of 1cm, and an LAI of 5
df <- data.frame(Tair=25,pressure=100,wind=c(3,4,5),ustar=c(0.5,0.6,0.65),H=c(200,230,250))
Gb.Su(data=df,zh=25,zr=40,d=17.5,Dl=0.01,LAI=5)

# the same meteorological conditions, but larger leaves
Gb.Su(data=df,zh=25,zr=40,d=17.5,Dl=0.1,LAI=5)

# same conditions, large leaves, and sparse canopy cover (LAI = 1.5)
Gb.Su(data=df,zh=25,zr=40,d=17.5,Dl=0.1,LAI=1.5)
```

Gb.Thom

Boundary Layer Conductance according to Thom 1972

Description

An empirical formulation for the canopy boundary layer conductance for heat transfer based on a simple ustar dependency.

Usage

```
Gb.Thom(ustar, Sc = NULL, Sc_name = NULL, constants = bigleaf.constants())
```

Arguments

ustar	Friction velocity (m s ⁻¹)
Sc	Optional: Schmidt number of additional quantities to be calculated
Sc_name	Optional: Name of the additional quantities, has to be of same length than Sc_name
constants	k - von-Karman constant Sc_CO2 - Schmidt number for CO2 Pr - Prandtl number (if Sc is provided)

Details

The empirical equation for Rb suggested by Thom 1972 is:

$$Rb = 6.2ustar^{-0.67}$$

Gb (=1/Rb) for water vapor and heat are assumed to be equal in this package. Gb for other quantities x is calculated as (Hicks et al. 1987):

$$Gb_x = Gb / (Sc_x / Pr)^{0.67}$$

where Sc_x is the Schmidt number of quantity x , and Pr is the Prandtl number (0.71).

Value

a data.frame with the following columns:

Gb_h	Boundary layer conductance for heat transfer (m s-1)
Rb_h	Boundary layer resistance for heat transfer (s m-1)
kB_h	kB-1 parameter for heat transfer
Gb_Sc_name	Boundary layer conductance for Sc_name (m s-1). Only added if Sc_name and Sc_name are provided

References

Thom, A., 1972: Momentum, mass and heat exchange of vegetation. Quarterly Journal of the Royal Meteorological Society 98, 124-134.

Hicks, B.B., Baldocchi, D.D., Meyers, T.P., Hosker, J.R., Matt, D.R., 1987: A preliminary multiple resistance routine for deriving dry deposition velocities from measured quantities. Water, Air, and Soil Pollution 36, 311-330.

See Also

[Gb.Choudhury](#), [Gb.Su](#), [aerodynamic.conductance](#)

Examples

```
Gb.Thom(seq(0.1, 1.4, 0.1))

## calculate Gb for SO2 as well
Gb.Thom(seq(0.1, 1.4, 0.1), Sc=1.25, Sc_name="SO2")
```

intercellular.CO2

Bulk Intercellular CO2 Concentration

Description

Bulk canopy intercellular CO2 concentration (C_i) calculated based on Fick's law given surface conductance (G_s), gross primary productivity (GPP) and atmospheric CO2 concentration (C_a).

Usage

```
intercellular.CO2(
  data,
  Ca = "Ca",
  GPP = "GPP",
  Gs = "Gs_mol",
  Rleaf = NULL,
  missing.Rleaf.as.NA = FALSE,
  constants = bigleaf.constants()
)
```

Arguments

data	Data.Frame or matrix with all required columns
Ca	Atmospheric or surface CO2 concentration (umol mol-1)
GPP	Gross primary productivity (umol CO2 m-2 s-1)
Gs	Surface conductance to water vapor (mol m-2 s-1)
Rleaf	Ecosystem respiration stemming from leaves (umol CO2 m-2 s-1); defaults to 0
missing.Rleaf.as.NA	if Rleaf is provided, should missing values be treated as NA (TRUE) or set to 0 (FALSE, the default)?
constants	DwDc - Ratio of the molecular diffusivities for water vapor and CO2 (-)

Details

Bulk intercellular CO2 concentration (C_i) is given by:

$$C_i = C_a - (GPP - Rleaf)/(Gs/1.6)$$

where $Gs/1.6$ (mol m-2 s-1) represents the surface conductance to CO2. Note that Gs is required in mol m-2 s-1 for water vapor. Gs is converted to its value for CO2 internally. C_a can either be atmospheric CO2 concentration (as measured), or surface CO2 concentration as calculated from [surface.CO2](#).

Value

C_i - Bulk canopy intercellular CO2 concentration (umol mol-1)

Note

The equation is based on Fick's law of diffusion and is equivalent to the often used equation at leaf level ($c_i = c_a - A_n/g_s$). Note that GPP and Gs have a different interpretation than A_n and g_s . Gs comprises non-physiological contributions (i.e. physical evaporation) and is confounded by physical factors (e.g. energy balance non-closure). GPP does not account for dark respiration and is further subject to uncertainties in the NEE partitioning algorithm used. Leaf respiration can be provided, but it is usually not known at ecosystem level (as a consequence, C_i is likely to be slightly underestimated) This function should be used with care and the resulting C_i might not be readily comparable to its leaf-level analogue and/or physiological meaningful.

References

Kosugi Y. et al., 2013: Determination of the gas exchange phenology in an evergreen coniferous forest from 7 years of eddy covariance flux data using an extended big-leaf analysis. *Ecol Res* 28, 373-385.

Keenan T., Sabate S., Gracia C., 2010: The importance of mesophyll conductance in regulating forest ecosystem productivity during drought periods. *Global Change Biology* 16, 1019-1034.

Examples

```
# calculate bulk canopy Ci of a productive ecosystem
intercellular.CO2(Ca=400,GPP=40,Gs=0.7)

# note the sign convention for NEE
```

isothermal.Rn

Isothermal Net Radiation

Description

Calculates the isothermal net radiation, i.e. the net radiation that the surface would receive if it had the same temperature than the air.

Usage

```
isothermal.Rn(
  data,
  Rn = "Rn",
  Tair = "Tair",
  Tsurf = "Tsurf",
  emissivity,
  constants = bigleaf.constants()
)
```

Arguments

data	Data.frame or matrix containing all required variables
Rn	Net radiation (W m ⁻²)
Tair	Air temperature (degC)
Tsurf	Surface temperature (degC)
emissivity	Emissivity of the surface (-)
constants	sigma - Stefan-Boltzmann constant (W m ⁻² K ⁻⁴) Kelvin - conversion degree Celsius to Kelvin

Details

The isothermal net radiation (Rni) is given by:

$$Rni = Rn + \epsilon * \sigma * (Tsurf^4 - Tair^4)$$

where ϵ is the emissivity of the surface. Tsurf and Tair are in Kelvin.

Value

Rni - isothermal net radiation (W m-2)

References

Jones, H. 2014: Plants and Microclimate. 3rd edition, Cambridge University Press.

Examples

```
# calculate isothermal net radiation of a surface that is 2degC warmer than the air.
isothermal.Rn(Rn=400,Tair=25,Tsurf=27,emissivity=0.98)
```

 kg.to.mol

Conversion between Mass and Molar Units

Description

Converts mass units of a substance to the corresponding molar units and vice versa.

Usage

```
kg.to.mol(mass, molarMass = bigleaf.constants()$H2Omol)
```

Arguments

mass Numeric vector of mass in kg
 molarMass Numeric vector of molar mass of the substance (kg mol-1) e.g. as provided by [bigleaf.constants\(\)](#)\$H2Omol Default is molar mass of Water.

Value

Numeric vector of amount of substance in mol.

kinematic.viscosity *Kinematic Viscosity of Air*

Description

calculates the kinematic viscosity of air.

Usage

```
kinematic.viscosity(Tair, pressure, constants = bigleaf.constants())
```

Arguments

Tair	Air temperature (deg C)
pressure	Atmospheric pressure (kPa)
constants	Kelvin - conversion degree Celsius to Kelvin pressure0 - reference atmospheric pressure at sea level (Pa) Tair0 - reference air temperature (K) kPa2Pa - conversion kilopascal (kPa) to pascal (Pa)

Details

where v is the kinematic viscosity of the air ($\text{m}^2 \text{s}^{-1}$), given by (Massman 1999b):

$$v = 1.327 * 10^{-5} (\text{pressure0}/\text{pressure}) (\text{Tair}/\text{Tair0})^{1.81}$$

Value

v - kinematic viscosity of air ($\text{m}^2 \text{s}^{-1}$)

References

Massman, W.J., 1999b: Molecular diffusivities of Hg vapor in air, O₂ and N₂ near STP and the kinematic viscosity and thermal diffusivity of air near STP. *Atmospheric Environment* 33, 453-457.

Examples

```
kinematic.viscosity(25,100)
```

`latent.heat.vaporization`*Latent Heat of Vaporization*

Description

Latent heat of vaporization as a function of air temperature.

Usage`latent.heat.vaporization(Tair)`**Arguments**

<code>Tair</code>	Air temperature (deg C)
-------------------	-------------------------

Details

The following formula is used:

$$\lambda = (2.501 - 0.00237 * T_{air})10^6$$

Value

λ	Latent heat of vaporization (J kg-1)
-----------	--------------------------------------

References

Stull, B., 1988: An Introduction to Boundary Layer Meteorology (p.641) Kluwer Academic Publishers, Dordrecht, Netherlands

Foken, T, 2008: Micrometeorology. Springer, Berlin, Germany.

Examples`latent.heat.vaporization(seq(5,45,5))`

LE.to.ET

Conversion between Latent Heat Flux and Evapotranspiration

Description

converts evaporative water flux from mass (ET=evapotranspiration) to energy (LE=latent heat flux) units, or vice versa.

Usage

LE.to.ET(LE, Tair)

ET.to.LE(ET, Tair)

Arguments

LE	Latent heat flux (W m-2)
Tair	Air temperature (deg C)
ET	Evapotranspiration (kg m-2 s-1)

Details

The conversions are given by:

$$ET = LE/\lambda$$

$$LE = \lambda ET$$

where λ is the latent heat of vaporization (J kg-1) as calculated by [latent.heat.vaporization](#).

Examples

```
# LE of 200 Wm-2 and air temperature of 25degC
LE.to.ET(200,25)
```

light.response	<i>Ecosystem Light Response</i>
----------------	---------------------------------

Description

calculates GPP_ref at a reference (usually saturating) PPFD and ecosystem quantum yield (alpha) using a rectangular light response curve.

Usage

```
light.response(
  data,
  NEE = "NEE",
  Reco = "Reco",
  PPFD = "PPFD",
  PPFD_ref = 2000,
  ...
)
```

Arguments

data	Data.frame or matrix containing all required columns
NEE	Net ecosystem exchange (umol CO2 m-2 s-1)
Reco	Ecosystem respiration (umol CO2 m-2 s-1)
PPFD	Photosynthetic photon flux density (umol m-2 s-1)
PPFD_ref	Reference PPFD (umol m-2 s-1) for which GPP_ref is estimated. Default is 2000 umol m-2 s-1.
...	Additional arguments to nls

Details

A rectangular light response curve is fitted to NEE data. The curve takes the form as described in Falge et al. 2001:

$$-NEE = \alpha PPFD / (1 - (PPFD / PPFD_{ref}) + \alpha PPFD / GPP_{ref}) - Reco$$

where α is the ecosystem quantum yield (umol CO2 m-2 s-1) (umol quanta m-2 s-1)-1, and GPP_ref is the GPP at the reference PPFD (usually at saturating light). α represents the slope of the light response curve, and is a measure for the light use efficiency of the canopy.

The advantage of this equation over the standard rectangular light response curve is that GPP_ref at PPFD_ref is more readily interpretable as it constitutes a value observed in the ecosystem, in contrast to GPP_ref (mostly named 'beta') in the standard model that occurs at infinite light. PPFD_ref defaults to 2000 umol m-2 s-1, but other values can be used. For further details refer to Falge et al. 2001.

Value

A nls model object containing estimates (+/- SE) for alpha and GPP_ref.

Note

Note the sign convention. Negative NEE indicates that carbon is taken up by the ecosystem. Reco has to be 0 or larger.

References

Falge E., et al. 2001: Gap filling strategies for defensible annual sums of net ecosystem exchange. *Agricultural and Forest Meteorology* 107, 43-69.

Gilmanov T.G., et al. 2003: Gross primary production and light response parameters of four Southern Plains ecosystems estimated using long-term CO₂-flux tower measurements. *Global Biogeochemical Cycles* 17, 1071.

Reichstein M., Stoy P.C., Desai A.R., Lasslop G., Richardson A. 2012: Partitioning of net fluxes. In: *Eddy Covariance. A practical guide to measurement and data analysis*. Aubinet M., Vesala T., Papale D. (Eds.). Springer.

light.use.encyclopedia *Light-Use Efficiency (LUE)*

Description

Amount of carbon fixed (GPP) per incoming light.

Usage

light.use.encyclopedia(GPP, PPFD)

Arguments

GPP	Gross ecosystem productivity (umol CO ₂ m ⁻² s ⁻¹)
PPFD	Photosynthetic photon flux density (umol quanta m ⁻² s ⁻¹)

Details

Light use efficiency is calculated as

$$LUE = \text{sum}(GPP) / \text{sum}(PPFD)$$

where both GPP and PPFD are in umol m⁻² s⁻¹. A more meaningful (as directly comparable across ecosystems) approach is to take absorbed PPFD rather than incoming PPFD as used here.

Value

LUE - Light use efficiency (-)

See Also[energy.use. efficiency](#)**Examples**

```
light.use. efficiency(GPP=20,PPFD=1500)
```

longwave.conductance *Longwave Radiative Transfer Conductance of the Canopy*

Description

Longwave Radiative Transfer Conductance of the Canopy

Usage

```
longwave.conductance(Tair, LAI, constants = bigleaf.constants())
```

Arguments

Tair	Air temperature (deg C)
LAI	Leaf area index (m ² m ⁻²)
constants	Kelvin - conversion degree Celsius to Kelvin sigma - Stefan-Boltzmann constant (W m ⁻² K ⁻⁴) cp - specific heat of air for constant pressure (J K ⁻¹ kg ⁻¹)

Details

the following formula is used (Martin, 1989):

$$Gr = 4\sigma T_{air}^3 LAI / cp$$

Value

Gr - longwave radiative transfer conductance of the canopy (m s⁻¹)

References

Martin P., 1989: The significance of radiative coupling between vegetation and the atmosphere. *Agricultural and Forest Meteorology* 49, 45-53.

Examples

```
longwave.conductance(25, seq(1, 8, 1))
```

Monin.Obukhov.length *Monin-Obukhov Length*

Description

calculates the Monin-Obukhov length.

Usage

```
Monin.Obukhov.length(
  data,
  Tair = "Tair",
  pressure = "pressure",
  ustar = "ustar",
  H = "H",
  constants = bigleaf.constants()
)
```

Arguments

data	Data.frame or matrix containing all required variables
Tair	Air temperature (deg C)
pressure	Atmospheric pressure (kPa)
ustar	Friction velocity (m s-1)
H	Sensible heat flux (W m-2)
constants	Kelvin - conversion degree Celsius to Kelvin cp - specific heat of air for constant pressure (J K-1 kg-1) k - von Karman constant (-) g - gravitational acceleration (m s-2)

Details

The Monin-Obukhov length (L) is given by:

$$L = -(\rho * cp * ustar^3 * Tair)/(k * g * H)$$

where *rho* is air density (kg m-3).

Value

L - Monin-Obukhov length (m)

Note

Note that L gets very small for very low ustar values with implications for subsequent functions using L as input. It is recommended to filter data and exclude low ustar values (ustar < ~0.2) beforehand.

References

Foken, T, 2008: Micrometeorology. Springer, Berlin, Germany.

See Also

[stability.parameter](#)

Examples

```
Monin.Obukhov.length(Tair=25,pressure=100,ustar=seq(0.2,1,0.1),H=seq(40,200,20))
```

ms.to.mol

Conversion between Conductance Units

Description

Converts conductances from mass (m s⁻¹) to molar units (mol m⁻² s⁻¹), or vice versa.

Usage

```
ms.to.mol(G_ms, Tair, pressure, constants = bigleaf.constants())
```

```
mol.to.ms(G_mol, Tair, pressure, constants = bigleaf.constants())
```

Arguments

G_ms	Conductance (m s ⁻¹)
Tair	Air temperature (deg C)
pressure	Atmospheric pressure (kPa)
constants	Kelvin - conversion degree Celsius to Kelvin Rgas - universal gas constant (J mol ⁻¹ K ⁻¹) kPa2Pa - conversion kilopascal (kPa) to pascal (Pa)
G_mol	Conductance (mol m ⁻² s ⁻¹)

Details

The conversions are given by:

$$G_{mol} = G_{ms} * pressure / (R_{gas} * T_{air})$$

$$G_{ms} = G_{mol} * (R_{gas} * T_{air}) / pressure$$

where Tair is in Kelvin and pressure in Pa (converted from kPa internally)

References

Jones, H.G. 1992. Plants and microclimate: a quantitative approach to environmental plant physiology. 2nd Edition., Cambridge University Press, Cambridge. 428 p

Examples

```
ms.to.mol(0.005,25,100)
```

optimum.temperature *Optimum temperature of Gross Primary Productivity*

Description

Calculates the relationship between Gross Primary Productivity (GPP) and Air Temperature (Tair) using boundary line analysis and derives the thermal optima. This function can also be used to find the boundary line relationship and optima of other variables such as NPP and NEP.

Usage

```
optimum.temperature(  
  data,  
  GPP = "GPP",  
  Tair = "Tair",  
  BLine = 0.9,  
  Obs_filter = 30  
)
```

Arguments

data	Dataframe containing the Gross Primary Productivity and Air Temperature observations
GPP	Name of column (in quotations, eg. "GPP") containing the Gross Primary Productivity observations (umol CO ₂ m ⁻² s ⁻¹).
Tair	Name of column (in quotations, eg. "Tair") containing the air temperature (degrees celcius) observations.
BLine	Quantile at which to place the boundary line in format "0.XX". Defaults to 0.90.
Obs_filter	Filter to remove air temperature bins with an insufficient number of observations. Defaults to 30.

Details

This function works by first binning GPP and air temperature observations to 1 degree temperature bins and then deriving the relationship between GPP and air temperature at a defined quantile using boundary line analysis. Observations are binned using a rounding function, so that each bin is centered on the degree integer value (eg. bin 18 contains values between 17.5 and 18.49). The boundary line is usually placed at the upper boundary of the distribution (see Webb 1972) however this functional allows the user to select any quantile, with the default of 0.9 selected for use with eddy covariance flux observations due to the high level of noise in these data (see Bennett et al, 2021). After binning observations, the function removes temperature bins with fewer observations than the default of 30 (this value can also be user defined). It then calculates the smoothed curve between GPP and air temperature using the loess function and derives the thermal optima of GPP (Topt). Topt is defined as the temperature bin at which GPP reaches its maximum along the smoothed boundary line.

Value

A list containing the following objects:

1. df.bl: A four column dataframe:
 - Tair_bin: air temperature bins in 1 degree increments
 - GPP_Bline: Value of GPP at the BLine
 - n_obs: number of observations in the air temperature bin
 - GPP_Bline_smooth: Value of GPP at the smoothed Bline
2. opt.temp: A named vector with two elements:
 - Topt: Thermal optima of GPP - the air temperature bin with maximum GPP along the smoothed Bline
 - GPP_bl: The boundary line GPP observation at Topt

References

- Bennett A. et al., 2021: Thermal optima of gross primary productivity are closely aligned with mean air temperatures across Australian wooded ecosystems. *Global Change Biology* 32(3), 280-293
- Webb, R. A. 1972. Use of the Boundary Line in the analysis of biological data. *Journal of Horticultural Science* 47, 309-319

Examples

```
# Locate the relationship between GPP and air temperature using default values
# for BLine and observation filter.

Gpp_ta <- optimum.temperature(data=AT_Neu_Jul_2010, GPP="GPP", Tair="Tair")

# Locate the relationship between GPP and air temperature at the 50th percentile,
# filtering temperature bins with fewer than 10 observations

Gpp_ta <- optimum.temperature(data=AT_Neu_Jul_2010,
                              GPP="GPP", Tair="Tair", BLine=0.50, Obs_filter=10)
```

photosynthetic.capacity

Bulk Canopy Photosynthetic Capacity (Vcmax and Jmax)

Description

Bulk canopy maximum carboxylation rate (Vcmax25), and maximum electron transport rate (Jmax25) at 25 degrees Celsius from bulk intercellular CO₂ concentration using the Farquhar et al. 1980 model for C3 photosynthesis.

Usage

```
photosynthetic.capacity(
  data,
  C3 = TRUE,
  Temp,
  GPP = "GPP",
  Ci,
  PPFD = "PPFD",
  PPFD_j = c(200, 500),
  PPFD_c = 1000,
  Rleaf = NULL,
  Oi = 0.21,
  Kc25 = 404.9,
  Ko25 = 278.4,
  Gam25 = 42.75,
  Kc_Ha = 79.43,
  Ko_Ha = 36.38,
  Gam_Ha = 37.83,
  Vcmax_Ha = 65.33,
  Vcmax_Hd = 200,
  Vcmax_dS = 0.635,
  Jmax_Ha = 43.9,
  Jmax_Hd = 200,
  Jmax_dS = 0.64,
  Theta = 0.7,
  alpha_canopy = 0.8,
  missing.Rleaf.as.NA = FALSE,
  Ci_C4 = 100,
  constants = bigleaf.constants()
)
```

Arguments

data	Data.Frame or matrix with all required columns
C3	C3 vegetation (TRUE, the default) or C4 vegetation (FALSE)?

Temp	Surface (or air) temperature (degC)
GPP	Gross primary productivity (umol m ⁻² s ⁻¹)
Ci	Bulk canopy intercellular CO ₂ concentration (umol mol ⁻¹)
PPFD	Photosynthetic photon flux density (umol m ⁻² s ⁻¹)
PPFD_j	PPFD threshold, below which the canopy is considered to be RuBP regeneration limited. Defaults to 500 umol m ⁻² s ⁻¹ .
PPFD_c	PPFD threshold, above which the canopy is considered to be Rubisco limited. Defaults to 1000 umol m ⁻² s ⁻¹ .
Rleaf	Ecosystem respiration stemming from leaves (umol CO ₂ m ⁻² s ⁻¹); defaults to 0
Oi	Intercellular O ₂ concentration (mol mol ⁻¹)
Kc25	Michaelis-Menten constant for CO ₂ at 25 degC (umol mol ⁻¹)
Ko25	Michaelis-Menten constant for O ₂ at 25 degC (mmol mol ⁻¹)
Gam25	Photorespiratory CO ₂ compensation point ('Gamma star') at 25 degC (umol mol ⁻¹)
Kc_Ha	Activation energy for Kc (kJ mol ⁻¹)
Ko_Ha	Activation energy for Ko (kJ mol ⁻¹)
Gam_Ha	Activation energy for Gam (kJ mol ⁻¹)
Vcmax_Ha	Activation energy for Vcmax (kJ mol ⁻¹)
Vcmax_Hd	Deactivation energy for Vcmax (kJ mol ⁻¹)
Vcmax_dS	Entropy term for Vcmax (kJ mol ⁻¹ K ⁻¹)
Jmax_Ha	Activation energy for Jmax (kJ mol ⁻¹)
Jmax_Hd	Deactivation energy for Jmax (kJ mol ⁻¹)
Jmax_dS	Entropy term for Jmax (kJ mol ⁻¹ K ⁻¹)
Theta	Curvature term in the light response function of J (-)
alpha_canopy	Canopy absorptance (-)
missing.Rleaf.as.NA	if Rleaf is provided, should missing values be treated as NA (TRUE) or set to 0 (FALSE, the default)?
Ci_C4	intercellular CO ₂ concentration below which photosynthesis is considered to be CO ₂ -limited (umol mol ⁻¹), ignored if C3 = TRUE.
constants	Kelvin - conversion degree Celsius to Kelvin Rgas - universal gas constant (J mol ⁻¹ K ⁻¹) kJ2J - conversion kilojoule (kJ) to joule (J) J2kJ - conversion joule (J) to kilojoule (kJ) se_median - conversion standard error (SE) of the mean to SE of the median

Details

The maximum carboxylation rate at 25degC (Vcmax25) and the maximum electron transport rate at 25degC (Jmax25), which characterize photosynthetic capacity, are calculated as at leaf level. The required variables Gs and Ci can be calculated from [surface.conductance](#) and [intercellular.CO2](#), respectively.

Gas exchange parameters are taken from Bernacchi et al. 2001 (apparent values, which assume an infinite mesophyll conductance). Negative and very low C_i values (the threshold is set to $C_i < 80 \mu\text{mol mol}^{-1}$ at the moment) are filtered out.

V_{cmax} is calculated from the photosynthesis model by Farquhar et al. 1980. If net photosynthesis is Rubisco-limited (RuBP-saturated carboxylation rate, i.e. light has to be (near-)saturating):

$$V_{cmax} = (GPP * (C_i + K_c * (1.0 + O_i/K_o))) / (C_i - G_{am})$$

where K_c and K_o are the Michaelis-Menten constants for CO_2 and O_2 (mmol mol^{-1}), respectively, O_i is the O_2 concentration, and G_{am} is the photorespiratory CO_2 compensation point ($\mu\text{mol mol}^{-1}$). Under low-light conditions, the electron transport rate J is calculated from the RuBP regeneration-limited photosynthesis rate:

$$J = (GPP * (4.0 * C_i + 8.0 * G_{am})) / (C_i - G_{am})$$

In this function, bulk canopy photosynthesis is assumed to be Rubisco/RuBP-regeneration limited, if incoming PPFD is above/below a specified threshold or range. These ranges are determined by the parameters $PPFD_j$ and $PPFD_c$. If, for example, $PPFD_j = c(100, 400)$, all conditions with a PPFD between 100 and 400 are assumed to be in the RuBP-regeneration (i.e. light-limited) photosynthesis domain. The electron transport rate J is then only calculated for periods that meet this criterion.

J_{max} is calculated from J and absorbed irradiance:

$$J = (APPFD_{PSII} + J_{max} - \sqrt{((APPFD_{PSII} + J_{max})^2 - 4.0 * Theta * APPFD_{PSII} * J_{max})}) / (2.0 * Theta)$$

where $APPFD_{PSII}$ is the absorbed PPFD by photosystem II (PS II), and $Theta$ is a curvature parameter. $APPFD_{PSII}$ is calculated as

$$PPFD * \alpha_{canopy} * 0.85 * \beta$$

where α_{canopy} is canopy-scale absorptance, 0.85 is a correction factor, and β is the fraction of photons absorbed by PS II (assumed 0.5). α_{canopy} accounts for non-absorbing components of the ecosystem such as stems or soil, and is very likely ecosystem-specific. This parameter is relatively sensitive for the determination of J_{max25} at some sites.

V_{cmax} and J_{max} at canopy level are assumed to follow the same temperature response as at leaf level. Hence, the respective parameter k at 25degC (k_{25}) is calculated as (see e.g. Kattge & Knorr 2007):

$$k_{25} = k / (\exp(H_a * (Temp - T_{ref}) / (T_{ref} * R_{gas} * Temp)) * (1 + \exp((T_{ref} * dS - H_d) / (T_{ref} * R_{gas})))) / (1 + \exp((Temp - T_{ref}) / (T_{ref} * R_{gas})))$$

where H_a is the activation energy (kJ mol^{-1}), H_d is the deactivation energy (kJ mol^{-1}), and dS is the entropy term ($\text{kJ mol}^{-1} \text{K}^{-1}$) of the respective parameter. T_{ref} is set to 298.15 K.

For C4 photosynthesis, the simplified model by von Caemmerer 2000 is used. For light-saturated photosynthesis, V_{cmax} is given by:

$$V_{cmax} = GPP$$

Note that in addition to the range PPF_{D_c}, the range C_{i_C4} discards all periods with low C_i, in which photosynthesis is likely to be CO₂-limited (see von Caemmerer 2000 for details).

In the light-limited case, J is calculated as:

$$J = 3 * GPP_j / (1 - 0.5)$$

The calculation of J_{max25} and V_{cmax25} is identical to C₃ photosynthesis as described above.

Value

a data.frame with the following columns:

V _{cmax25}	maximum bulk canopy carboxylation rate at 25degC (umol m ⁻² (ground) s ⁻¹)
J _{max25}	maximum bulk canopy electron transport rate at 25degC (umol m ⁻² (ground) s ⁻¹)

Note

The critical assumption is that bulk canopy photosynthesis is limited by one of the two limitation states. Incoming PPF_D is assumed to determine the limitation states. Note however that the ranges (PPF_{D_j} and PPF_{D_c}) are likely ecosystem-specific. E.g. dense canopies presumably require higher PPF_{D_c} thresholds than open canopies. A threshold of 500 umol m⁻² s⁻¹ PPF_D for Rubisco-limited photosynthesis was assumed a reasonable working assumption (see Kosugi et al. 2013). Here, PPF_{D_c} defaults to 1000 umol m⁻² s⁻¹. Note that even under very high/low irradiances, not all photosynthetically active plant material of an ecosystem will be in the same limitation state. Note that parameters describing bulk canopy photosynthetic capacity are not directly comparable to their leaf-level counterparts, as the former integrate over the entire canopy depth (i.e. are given per ground area, and not per leaf area). In general, the function should be used with care!

References

- Lloyd J. et al., 1995: A simple calibrated model of Amazon rainforest productivity based on leaf biochemical properties. *Plant, Cell and Environment* 18, 1129-1145.
- Rayment M.B., Loustau D., Jarvis P.G., 2002: Photosynthesis and respiration of black spruce at three organizational scales: shoot, branch and canopy. *Tree Physiology* 22, 219-229.
- Kosugi Y. et al., 2013: Determination of the gas exchange phenology in an evergreen coniferous forest from 7 years of eddy covariance flux data using an extended big-leaf analysis. *Ecol Res* 28, 373-385.
- Ueyama M. et al, 2016: Optimization of a biochemical model with eddy covariance measurements in black spruce forests of Alaska for estimating CO₂ fertilization effects. *Agricultural and Forest Meteorology* 222, 98-111.
- Bernacchi C.J., Singaas E.L., Pimentel C., Portis JR A.R., Long S.P., 2001: Improved temperature response functions for models of Rubisco-limited photosynthesis. *Plant, Cell and Environment* 24, 253-259.
- Bernacchi C.J., Pimentel C., Long S.P., 2003: In vivo temperature response functions of parameters required to model RuBP-limited photosynthesis. *Plant, Cell and Environment* 26, 1419-1430.
- von Caemmerer, 2000: Biochemical models of leaf photosynthesis. *Techniques in plant sciences* No. 2. CSIRO Publishing, Collingwood VIC, Australia.

See Also

[intercellular.CO2](#), [Arrhenius.temp.response](#)

Examples

```
DE_Tha_Jun_2014_2 <- filter.data(DE_Tha_Jun_2014,quality.control=FALSE,
                                vars.qc=c("Tair","precip","VPD","H","LE"),
                                filter.growseas=FALSE,filter.precip=TRUE,
                                filter.vars=c("Tair","PPFD","ustar","LE"),
                                filter.vals.min=c(5,200,0.2,0),
                                filter.vals.max=c(NA,NA,NA,NA),NA.as.invalid=TRUE,
                                quality.ext="_qc",good.quality=c(0,1),
                                missing.qc.as.bad=TRUE,GPP="GPP",doy="doy",
                                year="year",tGPP=0.5,ws=15,min.int=5,precip="precip",
                                tprecip=0.1,precip.hours=24,records.per.hour=2)

# calculate Ga
Ga <- aerodynamic.conductance(DE_Tha_Jun_2014_2,Rb_model="Thom_1972")["Ga_h"]

# calculate Gs from the the inverted PM equation
Gs_PM <- surface.conductance(DE_Tha_Jun_2014_2,Tair="Tair",pressure="pressure",
                              Rn="Rn",G="G",S=NULL,VPD="VPD",Ga=Ga,
                              formulation="Penman-Monteith")["Gs_mol"]

# calculate Ci
Ci <- intercellular.CO2(DE_Tha_Jun_2014_2,Ca="Ca",GPP="GPP",Gs=Gs_PM)

# calculate Vcmax25 and Jmax25
photosynthetic.capacity(DE_Tha_Jun_2014_2,Temp="Tair",Ci=Ci,PPFD_j=c(200,500),PPFD_c=1000)
```

potential.ET

Potential Evapotranspiration

Description

Potential evapotranspiration according to Priestley & Taylor 1972 or the Penman-Monteith equation with a prescribed surface conductance.

Usage

```
potential.ET(
  data,
  Tair = "Tair",
  pressure = "pressure",
  Rn = "Rn",
  G = NULL,
  S = NULL,
  VPD = "VPD",
```



```

Ga = "Ga_h",
approach = c("Priestley-Taylor", "Penman-Monteith"),
alpha = 1.26,
Gs_pot = 0.6,
missing.G.as.NA = FALSE,
missing.S.as.NA = FALSE,
Esat.formula = c("Sonntag_1990", "Alduchov_1996", "Allen_1998"),
constants = bigleaf.constants()
)

```

Arguments

data	Data.frame or matrix containing all required variables; optional
Tair	Air temperature (degC)
pressure	Atmospheric pressure (kPa)
Rn	Net radiation (W m-2)
G	Ground heat flux (W m-2); optional
S	Sum of all storage fluxes (W m-2); optional
VPD	Vapor pressure deficit (kPa); only used if approach = "Penman-Monteith".
Ga	Aerodynamic conductance to heat/water vapor (m s-1); only used if approach = "Penman-Monteith".
approach	Approach used. Either "Priestley-Taylor" (default), or "Penman-Monteith".
alpha	Priestley-Taylor coefficient; only used if approach = "Priestley-Taylor".
Gs_pot	Potential/maximum surface conductance (mol m-2 s-1); defaults to 0.6 mol m-2 s-1; only used if approach = "Penman-Monteith".
missing.G.as.NA	if TRUE, missing G are treated as NAs, otherwise set to 0.
missing.S.as.NA	if TRUE, missing S are treated as NAs, otherwise set to 0.
Esat.formula	Optional: formula to be used for the calculation of esat and the slope of esat. One of "Sonntag_1990" (Default), "Alduchov_1996", or "Allen_1998". See Esat.slope .
constants	cp - specific heat of air for constant pressure (J K-1 kg-1) eps - ratio of the molecular weight of water vapor to dry air Pa2kPa - conversion pascal (Pa) to kilopascal (kPa) Rd - gas constant of dry air (J kg-1 K-1) (only used if approach = "Penman-Monteith") Rgas - universal gas constant (J mol-1 K-1) (only used if approach = "Penman-Monteith") Kelvin - conversion degree Celsius to Kelvin (only used if approach = "Penman-Monteith")

Details

Potential evapotranspiration is calculated according to Priestley & Taylor, 1972 if approach = "Priestley-Taylor" (the default):

$$LE_{pot,PT} = (\alpha * \Delta * (Rn - G - S)) / (\Delta + \gamma)$$

α is the Priestley-Taylor coefficient, Δ is the slope of the saturation vapor pressure curve (kPa K-1), and γ is the psychrometric constant (kPa K-1). if approach = "Penman-Monteith", potential evapotranspiration is calculated according to the Penman-Monteith equation:

$$LE_{pot,PM} = (\Delta * (Rn - G - S) + \rho * cp * VPD * Ga) / (\Delta + \gamma * (1 + Ga/Gs_{pot}))$$

where Δ is the slope of the saturation vapor pressure curve (kPa K-1), ρ is the air density (kg m-3), and γ is the psychrometric constant (kPa K-1). The value of Gs_{pot} is typically a maximum value of Gs observed at the site, e.g. the 90th percentile of Gs within the growing season.

Value

a data.frame with the following columns:

ET_pot	Potential evapotranspiration (kg m-2 s-1)
LE_pot	Potential latent heat flux (W m-2)

Note

If the first argument data is provided (either a matrix or a data.frame), the following variables can be provided as character (in which case they are interpreted as the column name of data) or as numeric vectors, in which case they are taken directly for the calculations. If data is not provided, all input variables have to be numeric vectors.

References

Priestley, C.H.B., Taylor, R.J., 1972: On the assessment of surface heat flux and evaporation using large-scale parameters. Monthly Weather Review 100, 81-92.

Allen, R.G., Pereira L.S., Raes D., Smith M., 1998: Crop evapotranspiration - Guidelines for computing crop water requirements - FAO Irrigation and drainage paper 56.

Novick, K.A., et al. 2016: The increasing importance of atmospheric demand for ecosystem water and carbon fluxes. Nature Climate Change 6, 1023 - 1027.

See Also

[surface.conductance](#)

Examples

```
# Calculate potential ET of a surface that receives a net radiation of 500 Wm-2
# using Priestley-Taylor:
potential.ET(Tair=30,pressure=100,Rn=500,alpha=1.26,approach="Priestley-Taylor")

# Calculate potential ET for a surface with known Gs (0.5 mol m-2 s-1) and Ga (0.1 m s-1)
# using Penman-Monteith:
```

```
LE_pot_PM <- potential.ET(Gs_pot=0.5,Tair=20,pressure=100,VPD=2,Ga=0.1,Rn=400,
                          approach="Penman-Monteith")[, "LE_pot"]
LE_pot_PM

# now cross-check with the inverted equation
surface.conductance(Tair=20,pressure=100,VPD=2,Ga=0.1,Rn=400,LE=LE_pot_PM)
```

potential.radiation *Potential radiation*

Description

Compute potential radiation for given geolocation and day of year.

Usage

```
potential.radiation(doy, hour, latDeg, longDeg, timezone, useSolartime = TRUE)
```

Arguments

doy	Integer vector with day of year (start at 1), same length as hour or length 1.
hour	Numeric vector with daytime as decimal hour of local time zone
latDeg	Latitude (decimal degrees)
longDeg	Longitude (decimal degrees)
timezone	Time zone (hours)
useSolartime	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 to directly use local winter time.

Value

vector of potential radiation (W m⁻²)

Examples

```
hour <- seq(5, 18, by = 0.1)
potRadApparentLocal <- potential.radiation(
  160, hour, 39.94, -5.77, timezone = +1)
potRadTimezone <- potential.radiation(
  160, hour, 39.94, -5.77, timezone = +1, useSolartime = FALSE)
plot(potRadApparentLocal ~ hour, type = 'l'
     , ylab = 'potential radiation (W m-2)')
lines(potRadTimezone ~ hour, col = "blue")
abline(v = 12, col = "blue", lty = "dotted")
legend("bottomright", legend = c("solar time", "local winter time")
     , col = c("black", "blue"), inset = 0.05, lty = 1)
```

```
pressure.from.elevation
```

Atmospheric Pressure from Hypsometric Equation

Description

An estimate of mean pressure at a given elevation as predicted by the hypsometric equation.

Usage

```
pressure.from.elevation(  
  elev,  
  Tair,  
  VPD = NULL,  
  constants = bigleaf.constants()  
)
```

Arguments

elev	Elevation a.s.l. (m)
Tair	Air temperature (deg C)
VPD	Vapor pressure deficit (kPa); optional
constants	Kelvin- conversion degC to Kelvin pressure0 - reference atmospheric pressure at sea level (Pa) Rd - gas constant of dry air (J kg-1 K-1) g - gravitational acceleration (m s-2) Pa2kPa - conversion pascal (Pa) to kilopascal (kPa)

Details

Atmospheric pressure is approximated by the hypsometric equation:

$$pressure = pressure_0 / (\exp(g * elevation / (RdTemp)))$$

Value

pressure - Atmospheric pressure (kPa)

Note

The hypsometric equation gives an estimate of the standard pressure at a given altitude. If VPD is provided, humidity correction is applied and the virtual temperature instead of air temperature is used. VPD is internally converted to specific humidity.

References

Stull B., 1988: An Introduction to Boundary Layer Meteorology. Kluwer Academic Publishers, Dordrecht, Netherlands.

Examples

```
# mean pressure at 500m altitude at 25 deg C and VPD of 1 kPa
pressure.from.elevation(500,Tair=25,VPD=1)
```

```
psychrometric.constant
```

Psychrometric Constant

Description

Calculates the psychrometric 'constant'.

Usage

```
psychrometric.constant(Tair, pressure, constants = bigleaf.constants())
```

Arguments

Tair	Air temperature (deg C)
pressure	Atmospheric pressure (kPa)
constants	cp - specific heat of air for constant pressure (J K ⁻¹ kg ⁻¹) eps - ratio of the molecular weight of water vapor to dry air (-)

Details

The psychrometric constant (γ) is given as:

$$\gamma = cp * pressure / (eps * \lambda)$$

where λ is the latent heat of vaporization (J kg⁻¹), as calculated from [latent.heat.vaporization](#).

Value

γ - the psychrometric constant (kPa K⁻¹)

References

Monteith J.L., Unsworth M.H., 2008: Principles of Environmental Physics. 3rd Edition. Academic Press, London.

Examples

```
psychrometric.constant(seq(5,45,5),100)
```

 radiometric.surface.temp

Radiometric Surface Temperature

Description

Radiometric surface temperature from longwave radiation measurements.

Usage

```
radiometric.surface.temp(
  data,
  LW_up = "LW_up",
  LW_down = "LW_down",
  emissivity,
  constants = bigleaf.constants()
)
```

Arguments

data	Data.frame or matrix containing all required input variables
LW_up	Longwave upward radiation (W m-2)
LW_down	Longwave downward radiation (W m-2)
emissivity	Emissivity of the surface (-)
constants	sigma - Stefan-Boltzmann constant (W m-2 K-4) Kelvin - conversion degree Celsius to Kelvin

Details

Radiometric surface temperature ($Trad$) is calculated as:

$$Trad = ((LW_{up} - (1 - \epsilon) * LW_{down}) / (\sigma \epsilon))^{1/4}$$

Value

a data.frame with the following columns:

Trad_K	Radiometric surface temperature (K)
Trad_degC	Radiometric surface temperature (degC)

References

Wang, W., Liang, S., Meyers, T. 2008: Validating MODIS land surface temperature products using long-term nighttime ground measurements. *Remote Sensing of Environment* 112, 623-635.

Examples

```
# determine radiometric surface temperature for the site DE-Tha in June 2014
# assuming an emissivity of 0.98.
# (Note that variable 'LW_down' was only included for the DE-Tha example dataset
# and not for the others due restrictions on file size)
Trad <- radiometric.surface.temp(DE_Tha_Jun_2014,emissivity=0.98)
summary(Trad)
```

reference.ET	<i>Reference Evapotranspiration</i>
--------------	-------------------------------------

Description

Reference evapotranspiration calculated from the Penman-Monteith equation with a prescribed surface conductance. This function is deprecated. Use `potential.ET(...,approach="Penman-Monteith")` instead.

Usage

```
reference.ET(
  data,
  Gs_ref = 0.0143,
  Tair = "Tair",
  pressure = "pressure",
  VPD = "VPD",
  Rn = "Rn",
  Ga = "Ga_h",
  G = NULL,
  S = NULL,
  missing.G.as.NA = FALSE,
  missing.S.as.NA = FALSE,
  Esat.formula = c("Sonntag_1990", "Alduchov_1996", "Allen_1998"),
  constants = bigleaf.constants()
)
```

Arguments

data	Data.frame or matrix containing all required variables; optional
Gs_ref	Reference surface conductance (m s-1); defaults to 0.0143 m s-1.
Tair	Air temperature (degC)
pressure	Atmospheric pressure (kPa)
VPD	Vapor pressure deficit (kPa)
Rn	Net radiation (W m-2)
Ga	Aerodynamic conductance to heat/water vapor (m s-1)

G	Ground heat flux (W m ⁻²); optional
S	Sum of all storage fluxes (W m ⁻²); optional
missing.G.as.NA	if TRUE, missing G are treated as NAs, otherwise set to 0.
missing.S.as.NA	if TRUE, missing S are treated as NAs, otherwise set to 0.
Esat.formula	Optional: formula to be used for the calculation of esat and the slope of esat. One of "Sonntag_1990" (Default), "Alduchov_1996", or "Allen_1998". See Esat.slope .
constants	cp - specific heat of air for constant pressure (J K ⁻¹ kg ⁻¹) eps - ratio of the molecular weight of water vapor to dry air Rd - gas constant of dry air (J kg ⁻¹ K ⁻¹) (only if approach = "Penman-Monteith") Rgas - universal gas constant (J mol ⁻¹ K ⁻¹) (only if approach = "Penman-Monteith") Kelvin - conversion degree Celsius to Kelvin (only if approach = "Penman-Monteith")

Reynolds.Number	<i>Roughness Reynolds Number</i>
-----------------	----------------------------------

Description

calculates the Roughness Reynolds Number.

Usage

```
Reynolds.Number(Tair, pressure, ustar, z0m, constants = bigleaf.constants())
```

Arguments

Tair	Air temperature (deg C)
pressure	Atmospheric pressure (kPa)
ustar	Friction velocity (m s ⁻¹)
z0m	Roughness length (m)
constants	Kelvin - conversion degree Celsius to Kelvin pressure0 - reference atmospheric pressure at sea level (Pa) Tair0 - reference air temperature (K)

Details

The Roughness Reynolds Number is calculated as in Massman 1999a:

$$Re = z0m * ustar / \nu$$

where ν is the kinematic viscosity (m² s⁻¹).

Value

Re - Roughness Reynolds Number (-)

References

Massman, W.J., 1999a: A model study of kB H- 1 for vegetated surfaces using 'localized near-field' Lagrangian theory. *Journal of Hydrology* 223, 27-43.

Examples

Reynolds.Number(25,100,0.5,z0m=0.5)

Rg.to.PPFD	<i>Conversions between Global Radiation and Photosynthetic Photon Flux Density</i>
------------	--

Description

Converts radiation from W m-2 to umol m-2 s-1 and vice versa.

Usage

Rg.to.PPFD(Rg, J_to_mol = 4.6, frac_PAR = 0.5)

PPFD.to.Rg(PPFD, J_to_mol = 4.6, frac_PAR = 0.5)

Arguments

Rg	Global radiation = incoming short-wave radiation at the surface (W m-2)
J_to_mol	Conversion factor from J m-2 s-1 (= W m-2) to umol (quanta) m-2 s-1
frac_PAR	Fraction of incoming solar irradiance that is photosynthetically active radiation (PAR); defaults to 0.5
PPFD	Photosynthetic photon flux density (umol m-2 s-1)

Details

The conversion is given by:

$$PPFD = Rg * frac_{PAR} * J_{to_mol}$$

by default, the combined conversion factor (frac_PAR * J_to_mol) is 2.3

Examples

```
# convert a measured incoming short-wave radiation of 500 Wm-2 to
# PPFD in umol m-2 s-1 and backwards
Rg.to.PPFD(500)
PPFD.to.Rg(1150)
```

roughness.length.heat *Roughness length for heat*

Description

Roughness length for heat (thermal roughness length, z0h) from the kB-1 parameter and roughness length for momentum (z0m).

Usage

```
roughness.length.heat(z0m, kB_h)
```

Arguments

z0m	Roughness length for momentum (m)
kB_h	kB-1 parameter for heat transfer

Details

The roughness length for heat (z0h) can be calculated from the following relationship (e.g. Verma 1989):

$$kB_h = \ln(z0m/z0h)$$

it follows:

$$z0h = z0m/exp(kB_h)$$

Value

Roughness length for heat, z0h (m)

Note

If unknown, z0m can be calculated from [roughness.parameters](#). kB_h can be calculated from [Gb.Thom](#), [Gb.Choudhury](#), [Gb.Su](#) or [aerodynamic.conductance](#).

References

Verma, S., 1989: Aerodynamic resistances to transfers of heat, mass and momentum. In: Estimation of areal evapotranspiration, IAHS Pub, 177, 13-20.

Rigden, A., Li, D., Salvucci, G., 2018: Dependence of thermal roughness length on friction velocity across land cover types: A synthesis analysis using AmeriFlux data. Agricultural and Forest Meteorology 249, 512-519.

Examples

```
roughness.length.heat(2,2.5)
```

```
roughness.parameters  Roughness Parameters
```

Description

A simple approximation of the two roughness parameters displacement height (d) and roughness length for momentum (z0m).

Usage

```
roughness.parameters(
  method = c("canopy_height", "canopy_height&LAI", "wind_profile"),
  zh,
  frac_d = 0.7,
  frac_z0m = 0.1,
  LAI,
  zr,
  cd = 0.2,
  hs = 0.01,
  data,
  Tair = "Tair",
  pressure = "pressure",
  wind = "wind",
  ustar = "ustar",
  H = "H",
  d = NULL,
  z0m = NULL,
  stab_roughness = TRUE,
  stab_formulation = c("Dyer_1970", "Businger_1971"),
  constants = bigleaf.constants()
)
```

Arguments

method	Method to use, one of "canopy_height", "canopy_height&LAI", "wind_profile" NOTE: if method = "canopy_height", only the following three arguments are used. If method = "canopy_height&LAI", only zh, LAI, cd, and hs are required.
zh	Vegetation height (m)
frac_d	Fraction of displacement height on canopy height (-)
frac_z0m	Fraction of roughness length on canopy height (-)
LAI	Leaf area index (-)
zr	Instrument (reference) height (m)
cd	Mean drag coefficient for individual leaves. Defaults to 0.2. Only needed if method = "canopy_height&LAI".
hs	roughness length of the soil surface (m). Only needed if method = "canopy_height&LAI" The following arguments are only needed if method = "wind_profile"!
data	Data.frame or matrix containing all required variables
Tair	Air temperature (deg C)
pressure	Atmospheric pressure (kPa)
wind	Wind speed at height zr (m s-1)
ustar	Friction velocity (m s-1)
H	Sensible heat flux (W m-2)
d	Zero-plane displacement height (m); optional
z0m	Roughness length for momentum (m); optional
stab_roughness	Should stability correction be considered? Default is TRUE.
stab_formulation	Stability correction function used (If stab_correction = TRUE). Either "Dyer_1970" or "Businger_1971".
constants	k - von-Karman constant (-) Kelvin - conversion degree Celsius to Kelvin cp - specific heat of air for constant pressure (J K-1 kg-1) g - gravitational acceleration (m s-2) se_median - conversion standard error (SE) of the mean to SE of the median

Details

The two main roughness parameters, the displacement height (d) and the roughness length for momentum (z0m) can be estimated from simple empirical relationships with canopy height (zh). If method = "canopy_height", the following formulas are used:

$$d = frac_d * zh$$

$$z0m = frac_z0m * zh$$

where `frac_d` defaults to 0.7 and `frac_z0m` to 0.1.

Alternatively, `d` and `z0m` can be estimated from both canopy height and LAI (If `method = "canopy_height&LAI"`). Based on data from Shaw & Pereira 1982, Choudhury & Monteith 1988 proposed the following semi-empirical relations:

$$X = cd * LAI$$

$$d = 1.1 * zh * \ln(1 + X^{1/4})$$

$$z0m = hs + 0.3 * zh * X^{1/2} \text{ for } 0 \leq X \leq 0.2$$

$$z0m = hs * zh * (1 - d/zh) \text{ for } 0.2 < X$$

If `method = "wind_profile"`, `z0m` is estimated by solving the wind speed profile for `z0m`:

$$z0m = \text{median}((zr - d) * \exp(-k * wind/ustar - psi_m))$$

By default, `d` in this equation is fixed to `0.7*zh`, but can be set to any other value. `psi_m` is 0 if `stab_roughness = FALSE`.

Value

a data.frame with the following columns:

<code>d</code>	Zero-plane displacement height (m)
<code>z0m</code>	Roughness length for momentum (m)
<code>z0m_se</code>	Only if <code>method = wind_profile</code> : Standard Error of the median for <code>z0m</code> (m)

References

Choudhury, B. J., Monteith J.L., 1988: A four-layer model for the heat budget of homogeneous land surfaces. Q. J. R. Meteorol. Soc. 114, 373-398.

Shaw, R. H., Pereira, A., 1982: Aerodynamic roughness of a plant canopy: a numerical experiment. Agricultural Meteorology, 26, 51-65.

See Also

[wind.profile](#)

Examples

```
# estimate d and z0m from canopy height for a dense (LAI=5) and open (LAI=2) canopy
roughness.parameters(method="canopy_height&LAI",zh=25,LAI=5)
roughness.parameters(method="canopy_height&LAI",zh=25,LAI=2)

# fix d to 0.7*zh and estimate z0m from the wind profile
df <- data.frame(Tair=c(25,25,25),pressure=100,wind=c(3,4,5),ustar=c(0.5,0.6,0.65),H=200)
roughness.parameters(method="wind_profile",zh=25,zr=40,frac_d=0.7,data=df)

# assume d = 0.8*zh
roughness.parameters(method="wind_profile",zh=25,zr=40,frac_d=0.8,data=df)
```

stability.correction *Integrated Stability Correction Functions for Heat and Momentum*

Description

dimensionless stability functions needed to correct deviations from the exponential wind profile under non-neutral conditions.

Usage

```
stability.correction(zeta, formulation = c("Dyer_1970", "Businger_1971"))
```

Arguments

zeta	Stability parameter zeta (-)
formulation	Formulation for the stability function. Either "Dyer_1970", or "Businger_1971"

Details

The functions give the integrated form of the universal functions. They depend on the value of the stability parameter ζ , which can be calculated from the function [stability.parameter](#). The integration of the universal functions is:

$$\psi = -x * zeta$$

for stable atmospheric conditions ($\zeta \geq 0$), and

$$\psi = 2 * \log((1 + y)/2)$$

for unstable atmospheric conditions ($\zeta < 0$).

The different formulations differ in their value of x and y.

Value

a data.frame with the following columns:

psi_h	the value of the stability function for heat and water vapor (-)
psi_m	the value of the stability function for momentum (-)

References

- Dyer, A.J., 1974: A review of flux-profile relationships. *Boundary-Layer Meteorology* 7, 363-372.
- Dyer, A. J., Hicks, B.B., 1970: Flux-Gradient relationships in the constant flux layer. *Quart. J. R. Meteorol. Soc.* 96, 715-721.
- Businger, J.A., Wyngaard, J. C., Izumi, I., Bradley, E. F., 1971: Flux-Profile relationships in the atmospheric surface layer. *J. Atmospheric Sci.* 28, 181-189.
- Paulson, C.A., 1970: The mathematical representation of wind speed and temperature profiles in the unstable atmospheric surface layer. *Journal of Applied Meteorology* 9, 857-861.
- Foken, T, 2008: *Micrometeorology*. Springer, Berlin, Germany.

Examples

```
zeta <- seq(-2,0.5,0.05)
stability.correction(zeta)
stability.correction(zeta,formulation="Businger_1971")
```

stability.parameter *Stability Parameter "zeta"*

Description

calculates "zeta", a parameter characterizing stratification in the lower atmosphere.

Usage

```
stability.parameter(
  data,
  Tair = "Tair",
  pressure = "pressure",
  ustar = "ustar",
  H = "H",
  zr,
  d,
  constants = bigleaf.constants()
)
```

Arguments

data	Data.frame or matrix containing all required variables
Tair	Air temperature (degC)
pressure	Atmospheric pressure (kPa)
ustar	Friction velocity (m s-1)
H	Sensible heat flux (W m-2)
zr	Instrument (reference) height (m)
d	Zero-plane displacement height (m)
constants	Kelvin - conversion degree Celsius to Kelvin cp - specific heat of air for constant pressure (J K-1 kg-1) k - von Karman constant (-) g - gravitational acceleration (m s-2)

Details

The stability parameter ζ is given by:

$$\zeta = (zr - d)/L$$

where L is the Monin-Obukhov length (m), calculated from the function [Monin.Obukhov.length](#). The displacement height d can be estimated from the function [roughness.parameters](#).

Value

ζ - stability parameter (-)

Examples

```
df <- data.frame(Tair=25,pressure=100,ustar=seq(0.2,1,0.1),H=seq(40,200,20))
stability.parameter(df,zr=40,d=15)
```

stomatal.sensitivity *Stomatal Sensitivity to VPD*

Description

Sensitivity of surface conductance to vapor pressure deficit.

Usage

```
stomatal.sensitivity(data, Gs = "Gs_mol", VPD = "VPD", ...)
```


Arguments

data	Data.frame or matrix containing all required columns
Gs	Surface conductance to water vapor (mol m ⁻² s ⁻¹)
VPD	Vapor pressure deficit (kPa)
...	Additional arguments to <code>nls</code>

Details

The function fits the following equation (Oren et al. 1999):

$$Gs = -m \ln(VPD) + b$$

where b is the reference surface conductance (Gs) at VPD=1kPa (in mol m⁻² s⁻¹), and m is the sensitivity parameter of Gs to VPD (in mol m⁻² s⁻¹ log(kPa)⁻¹). The two parameters b and m are fitted using `nls`. VPD can be the one directly measured at instrument height, or the one at the surface, as returned by `surface.conditions`.

Value

A `nls` model object containing (amongst others) estimates for the mean and standard errors of the parameters m and b.

References

- Oren R., et al. 1999: Survey and synthesis of intra- and interspecific variation in stomatal sensitivity to vapour pressure deficit. *Plant, Cell & Environment* 22, 1515-1526.
- Novick K.A., et al. 2016: The increasing importance of atmospheric demand for ecosystem water and carbon fluxes. *Nature Climate Change* 6, 1023 - 1027.

See Also

[surface.conductance](#)

Examples

```
## calculate Ga, Gs, and the stomatal sensitivity to VPD for the site FR-Pue in
## May 2012. Data are filtered for daytime, sufficiently high ustar, etc.
FR_Pue_May_2012_2 <- filter.data(FR_Pue_May_2012,quality.control=TRUE,
                                vars.qc=c("Tair","precip","H","LE"),
                                filter.growseas=FALSE,filter.precip=TRUE,
                                filter.vars=c("Tair","PPFD","ustar","VPD"),
                                filter.vals.min=c(5,200,0.2,0.3),
                                filter.vals.max=c(NA,NA,NA,NA),
                                NA.as.invalid=TRUE,quality.ext="_qc",
                                good.quality=c(0,1),missing.qc.as.bad=TRUE,
                                precip="precip",tprecip=0.1,precip.hours=24,
                                records.per.hour=2)
Ga <- aerodynamic.conductance(FR_Pue_May_2012_2)
Gs <- surface.conductance(FR_Pue_May_2012_2,Ga=Ga[, "Ga_h"])
```

```
stomatal.sensitivity(FR_Pue_May_2012_2,Gs=Gs[,"Gs_mol"],VPD="VPD")
```

```
stomatal.slope      Stomatal Slope Parameter "g1"
```

Description

Estimation of the intrinsic WUE metric "g1" (stomatal slope) from nonlinear regression.

Usage

```
stomatal.slope(
  data,
  Tair = "Tair",
  pressure = "pressure",
  GPP = "GPP",
  Gs = "Gs_mol",
  VPD = "VPD",
  Ca = "Ca",
  Rleaf = NULL,
  model = c("US0", "Ball&Berry", "Leuning"),
  robust.nls = FALSE,
  nmin = 40,
  fitg0 = FALSE,
  g0 = 0,
  fitD0 = FALSE,
  D0 = 1.5,
  Gamma = 50,
  missing.Rleaf.as.NA = FALSE,
  constants = bigleaf.constants(),
  ...
)
```

Arguments

data	Data.frame or matrix containing all required columns
Tair	Air (or surface) temperature (deg C)
pressure	Atmospheric pressure (kPa)
GPP	Gross primary productivity (umol CO2 m-2 s-1)
Gs	Surface conductance to water vapor (mol m-2 s-1)
VPD	Vapor pressure deficit (kPa)
Ca	Atmospheric CO2 concentration (air or surface) (umol mol-1)
Rleaf	Ecosystem respiration stemming from leaves (umol CO2 m-2 s-1); defaults to 0
model	Stomatal model used. One of "US0", "Ball&Berry", "Leuning".

<code>robust.nls</code>	Use robust nonlinear regression (<code>nlrob</code>)? Default is FALSE.
<code>nmin</code>	Minimum number of data required to perform the fit; defaults to 40.
<code>fitg0</code>	Should <code>g0</code> and <code>g1</code> be fitted simultaneously?
<code>g0</code>	Minimum stomatal conductance (mol m ⁻² s ⁻¹); ignored if <code>fitg0 = TRUE</code> .
<code>fitD0</code>	Should <code>D0</code> be fitted along with <code>g1</code> (and <code>g0</code> if <code>fitg0 = TRUE</code>)?; only used if <code>model = "Leuning"</code> .
<code>D0</code>	Stomatal sensitivity parameter to VPD; only used if <code>model = "Leuning"</code> and <code>fitD0 = FALSE</code> .
<code>Gamma</code>	Canopy CO ₂ compensation point (umol mol ⁻¹); only used if <code>model = "Leuning"</code> . Can be a constant or a variable. Defaults to 50 umol mol ⁻¹ .
<code>missing.Rleaf.as.NA</code>	if <code>Rleaf</code> is provided, should missing values be treated as NA (TRUE) or set to 0 (FALSE, the default)?
<code>constants</code>	Kelvin - conversion degree Celsius to Kelvin Rgas - universal gas constant (J mol ⁻¹ K ⁻¹) DwDc - Ratio of the molecular diffusivities for water vapor and CO ₂
<code>...</code>	Additional arguments to <code>nls</code> or <code>nlrob</code> if <code>robust.nls = TRUE</code> .

Details

All stomatal models were developed at leaf-level, but its parameters can also be estimated at ecosystem level (but be aware of caveats).

The unified stomatal optimization (USO) model is given by (Medlyn et al. 2011):

$$gs = g0 + 1.6 * (1.0 + g1/sqrt(VPD)) * An/ca$$

The semi-empirical model by Ball et al. 1987 is defined as:

$$gs = g0 + g1 * ((An * rH)/ca)$$

Leuning 1995 suggested a revised version of the Ball&Berry model:

$$gs = g0 + g1 * An/((ca - \Gamma) * (1 + VPD/D0))$$

where Γ is by default assumed to be constant, but likely varies with temperature and among plant species. The equations above are valid at leaf-level. At ecosystem level, An is replaced by GPP (or GPP - Rleaf, where Rleaf is leaf respiration), and gs (stomatal conductance) by G_s (surface conductance). The parameters in the models are estimated using nonlinear regression (`nls`) if `robust.nls = FALSE` and weighted nonlinear regression if `robust.nls = TRUE`. The weights are calculated from `nlrob`, and `nls` is used for the actual fitting. Alternatively to measured VPD and Ca (i.e. conditions at instrument height), conditions at the big-leaf surface can be provided. Those can be calculated using `surface.conditions`.

Value

A `nls` model object, containing information on the fitted parameters, their uncertainty range, model fit, etc.

References

- Medlyn B.E., et al., 2011: Reconciling the optimal and empirical approaches to modelling stomatal conductance. *Global Change Biology* 17, 2134-2144.
- Ball T.J., Woodrow I.E., Berry J.A. 1987: A model predicting stomatal conductance and its contribution to the control of photosynthesis under different environmental conditions. In: *Progress in Photosynthesis Research*, edited by J. Biggins, pp. 221-224, Martinus Nijhoff Publishers, Dordrecht, Netherlands.
- Leuning R., 1995: A critical appraisal of a combined stomatal-photosynthesis model for C3 plants. *Plant, Cell and Environment* 18, 339-355.
- Knauer, J. et al., 2018: Towards physiologically meaningful water-use efficiency estimates from eddy covariance data. *Global Change Biology* 24, 694-710.

See Also

[surface.conductance](#)

Examples

```
## filter data to ensure that Gs is a meaningful proxy to canopy conductance (Gc)
DE_Tha_Jun_2014_2 <- filter.data(DE_Tha_Jun_2014,quality.control=FALSE,
                                vars.qc=c("Tair","precip","VPD","H","LE"),
                                filter.growseas=FALSE,filter.precip=TRUE,
                                filter.vars=c("Tair","PPFD","ustar","LE"),
                                filter.vals.min=c(5,200,0.2,0),
                                filter.vals.max=c(NA,NA,NA,NA),NA.as.invalid=TRUE,
                                quality.ext="_qc",good.quality=c(0,1),
                                missing.qc.as.bad=TRUE,GPP="GPP",doy="doy",
                                year="year",tGPP=0.5,ws=15,min.int=5,precip="precip",
                                tprecip=0.1,precip.hours=24,records.per.hour=2)

# calculate Gs from the the inverted PM equation
Ga <- aerodynamic.conductance(DE_Tha_Jun_2014_2,Rb_model="Thom_1972"),["Ga_h"]

# if G and/or S are available, don't forget to indicate (they are ignored by default).
Gs_PM <- surface.conductance(DE_Tha_Jun_2014_2,Tair="Tair",pressure="pressure",
                              Rn="Rn",G="G",S=NULL,VPD="VPD",Ga=Ga,
                              formulation="Penman-Monteith"),["Gs_mol"]

### Estimate the stomatal slope parameter g1 using the USO model
mod_USO <- stomatal.slope(DE_Tha_Jun_2014_2,model="USO",GPP="GPP",Gs=Gs_PM,
                          robust.nls=FALSE,nmin=40,fitg0=FALSE)

### Use robust regression to minimize influence of outliers in Gs
mod_USO <- stomatal.slope(DE_Tha_Jun_2014_2,model="USO",GPP="GPP",Gs=Gs_PM,
                          robust.nls=TRUE,nmin=40,fitg0=FALSE)

### Estimate the same parameter from the Ball&Berry model and prescribe g0
mod_BB <- stomatal.slope(DE_Tha_Jun_2014_2,model="Ball&Berry",GPP="GPP",
                          robust.nls=FALSE,Gs=Gs_PM,g0=0.01,nmin=40,fitg0=FALSE)
```

```
## same for the Leuning model, but this time estimate both g1 and g0 (but fix D0)
mod_Leu <- stomatal.slope(DE_Tha_Jun_2014_2,model="Leuning",GPP="GPP",Gs=G_s_PM,
  robust.nls=FALSE,nmin=40,fitg0=FALSE,D0=1.5,fitD0=FALSE)
```

 surface.CO2

CO2 Concentration at the Canopy Surface

Description

the CO2 concentration at the canopy surface derived from net ecosystem CO2 exchange and measured atmospheric CO2 concentration.

Usage

```
surface.CO2(Ca, NEE, Ga_CO2, Tair, pressure)
```

Arguments

Ca	Atmospheric CO2 concentration (umol mol-1)
NEE	Net ecosystem exchange (umol CO2 m-2 s-1)
Ga_CO2	Aerodynamic conductance for CO2 (m s-1)
Tair	Air temperature (degC)
pressure	Atmospheric pressure (kPa)

Details

CO2 concentration at the canopy surface is calculated as:

$$C_{a_{surf}} = C_a + NEE/Ga_{CO2}$$

Note that this equation can be used for any gas measured (with NEE replaced by the net exchange of the respective gas and Ga_CO2 by the Ga of that gas).

Value

Ca_surf - CO2 concentration at the canopy surface (umol mol-1)

Note

the following sign convention is employed: negative values of NEE denote net CO2 uptake by the ecosystem.

Examples

```
surface.CO2(Ca=400,NEE=-30,Ga_CO2=0.05,Tair=25,pressure=100)
```

surface.conditions *Big-Leaf Surface Conditions*

Description

Calculates meteorological conditions at the big-leaf surface by inverting bulk transfer equations for water, energy, and carbon fluxes.

Usage

```
surface.conditions(
  data,
  Tair = "Tair",
  pressure = "pressure",
  LE = "LE",
  H = "H",
  VPD = "VPD",
  Ga = "Ga_h",
  calc.surface.CO2 = FALSE,
  Ca = "Ca",
  Ga_CO2 = "Ga_CO2",
  NEE = "NEE",
  Esat.formula = c("Sonntag_1990", "Alduchov_1996", "Allen_1998"),
  constants = bigleaf.constants()
)
```

Arguments

data	Data.frame or matrix containing all required input variables
Tair	Air temperature (deg C)
pressure	Atmospheric pressure (kPa)
LE	Latent heat flux (W m-2)
H	Sensible heat flux (W m-2)
VPD	Vapor pressure deficit (kPa)
Ga	Aerodynamic conductance for heat/water vapor (m s-1)
calc.surface.CO2	Calculate surface CO2 concentration? Defaults to FALSE.
Ca	Atmospheric CO2 concentration (mol mol-1). Required if calc.surface.CO2 = TRUE.
Ga_CO2	Aerodynamic conductance for CO2 (m s-1). Required if calc.surface.CO2 = TRUE.
NEE	Net ecosystem exchange (umol m-2 s-1). Required if calc.surface.CO2 = TRUE.

Esat.formula	Optional: formula to be used for the calculation of esat and the slope of esat. One of "Sonntag_1990" (Default), "Alduchov_1996", or "Allen_1998". See Esat.slope .
constants	cp - specific heat of air for constant pressure (J K-1 kg-1) eps - ratio of the molecular weight of water vapor to dry air (-) Pa2kPa - conversion pascal (Pa) to kilopascal (kPa)

Details

Canopy surface temperature and humidity are calculated by inverting bulk transfer equations of sensible and latent heat, respectively. 'Canopy surface' in this case refers to the surface of the big-leaf (i.e. at height $d + z0h$; the apparent sink of sensible heat and water vapor). Aerodynamic canopy surface temperature is given by:

$$T_{surf} = T_{air} + H / (\rho * cp * Ga)$$

where ρ is air density (kg m-3). Vapor pressure at the canopy surface is:

$$e_{surf} = e + (LE * \gamma) / (Ga * \rho * cp)$$

where γ is the psychrometric constant (kPa K-1). Vapor pressure deficit (VPD) at the canopy surface is calculated as:

$$VPD_{surf} = Esat_{surf} - e_{surf}$$

CO2 concentration at the canopy surface is given by:

$$Ca_{surf} = Ca + NEE / Ga_{CO2}$$

Note that Ga is assumed to be equal for water vapor and sensible heat. Ga is further assumed to be the inverse of the sum of the turbulent part and the canopy boundary layer conductance ($1/Ga = 1/Ga_m + 1/Gb$; see [aerodynamic.conductance](#)). Ga_{CO2} , the aerodynamic conductance for CO2 is also calculated by [aerodynamic.conductance](#). If Ga is replaced by Ga_m (i.e. only the turbulent conductance part), the results of the functions represent conditions outside the canopy boundary layer, i.e. in the canopy airspace.

Value

a data.frame with the following columns:

Tsurf	Surface temperature (deg C)
esat_surf	Saturation vapor pressure at the surface (kPa)
esurf	vapor pressure at the surface (kPa)

VPD_surf	vapor pressure deficit at the surface (kPa)
qsurf	specific humidity at the surface (kg kg ⁻¹)
rH_surf	relative humidity at the surface (-)
Ca_surf	CO ₂ concentration at the surface (umol mol ⁻¹)

Note

The following sign convention for NEE is employed (relevant if `calc.surface.CO2 = TRUE`): negative values of NEE denote net CO₂ uptake by the ecosystem.

References

- Knauer, J. et al., 2018: Towards physiologically meaningful water-use efficiency estimates from eddy covariance data. *Global Change Biology* 24, 694-710.
- Blanken, P.D. & Black, T.A., 2004: The canopy conductance of a boreal aspen forest, Prince Albert National Park, Canada. *Hydrological Processes* 18, 1561-1578.
- Shuttleworth, W. J., Wallace, J.S., 1985: Evaporation from sparse crops- an energy combination theory. *Quart. J. R. Met. Soc.* 111, 839-855.

Examples

```
# calculate surface temperature, water vapor, VPD etc. at the surface
# for a given temperature and turbulent fluxes, and under different
# aerodynamic conductance.
surface.conditions(Tair=25,pressure=100,LE=100,H=200,VPD=1.2,Ga=c(0.02,0.05,0.1))

# now calculate also surface CO2 concentration
surface.conditions(Tair=25,pressure=100,LE=100,H=200,VPD=1.2,Ga=c(0.02,0.05,0.1),
                  Ca=400,Ga_CO2=c(0.02,0.05,0.1),NEE=-20,calc.surface.CO2=TRUE)
```

surface.conductance *Surface Conductance to Water Vapor*

Description

Calculates surface conductance to water vapor from the inverted Penman-Monteith equation (by default) or from a simple flux-gradient approach.

Usage

```

surface.conductance(
  data,
  Tair = "Tair",
  pressure = "pressure",
  Rn = "Rn",
  G = NULL,
  S = NULL,
  VPD = "VPD",
  LE = "LE",
  Ga = "Ga_h",
  missing.G.as.NA = FALSE,
  missing.S.as.NA = FALSE,
  formulation = c("Penman-Monteith", "Flux-Gradient"),
  Esat.formula = c("Sonntag_1990", "Alduchov_1996", "Allen_1998"),
  constants = bigleaf.constants()
)

```

Arguments

data	Data.frame or matrix containing all required input variables
Tair	Air temperature (deg C)
pressure	Atmospheric pressure (kPa)
Rn	Net radiation (W m-2)
G	Ground heat flux (W m-2); optional
S	Sum of all storage fluxes (W m-2); optional
VPD	Vapor pressure deficit (kPa)
LE	Latent heat flux (W m-2)
Ga	Aerodynamic conductance to heat/water vapor (m s-1)
missing.G.as.NA	if TRUE, missing G are treated as NAs, otherwise they are set to 0. Only used if formulation = "Penman-Monteith".
missing.S.as.NA	if TRUE, missing S are treated as NAs, otherwise they are set to 0. Only used if formulation = "Penman-Monteith".
formulation	Formulation used. Either "Penman-Monteith" (the default) using the inverted Penman-Monteith equation, or "Flux-Gradient", for a simple flux-gradient approach requiring ET, pressure, and VPD only.
Esat.formula	Optional: formula to be used for the calculation of esat and the slope of esat. One of "Sonntag_1990" (Default), "Alduchov_1996", or "Allen_1998". Only used if formulation = "Penman-Monteith". See Esat.slope .
constants	cp - specific heat of air for constant pressure (J K-1 kg-1) eps - ratio of the molecular weight of water vapor to dry air (-) Rd - gas constant of dry air (J kg-1 K-1) Rgas - universal gas constant (J mol-1 K-1)

Kelvin - conversion degree Celsius to Kelvin
 Mw - molar mass of water vapor (kg mol-1)
 Pa2kPa - conversion pascal (Pa) to kilopascal (kPa)

Details

If `formulation = "Penman-Monteith"` (the default), surface conductance (Gs) in m s-1 is calculated from the inverted Penman-Monteith equation:

$$Gs = (LE * Ga * \gamma) / (\Delta * A + \rho * cp * Ga * VPD - LE * (\Delta + \gamma))$$

Where γ is the psychrometric constant (kPa K-1), Δ is the slope of the saturation vapor pressure curve (kPa K-1), and ρ is air density (kg m-3). Available energy (A) is defined as $A = Rn - G - S$. If G and/or S are not provided, $A = Rn$.

By default, any missing data in G and S are set to 0. If `missing.S.as.NA = TRUE` or `missing.S.as.NA = TRUE`, Gs will give NA for these timesteps.

If `formulation="Flux-Gradient"`, Gs (in mol m-2 s-1) is calculated from VPD and ET only:

$$Gs = ET / pressure * VPD$$

where ET is in mol m-2 s-1. Note that this formulation assumes fully coupled conditions (i.e. $G_a = \infty$). This formulation is equivalent to the inverted form of Eq.6 in McNaughton & Black 1973:

$$Gs = LE * \gamma / (\rho * cp * VPD)$$

which gives Gs in m s-1. Note that $G_s > G_c$ (canopy conductance) under conditions when a significant fraction of ET comes from interception or soil evaporation.

If pressure is not available, it can be approximated by elevation using the function [pressure.from.elevation](#)

Value

a dataframe with the following columns:

Gs_ms	Surface conductance in m s-1
Gs_mol	Surface conductance in mol m-2 s-1

References

Monteith, J., 1965: Evaporation and environment. In Fogg, G. E. (Ed.), The state and movement of water in living organisms (pp.205-234). 19th Symp. Soc. Exp. Biol., Cambridge University Press, Cambridge

McNaughton, K.G., Black, T.A., 1973: A study of evapotranspiration from a Douglas Fir forest using the energy balance approach. Water Resources Research 9, 1579-1590.

Examples

```
## filter data to ensure that Gs is a meaningful proxy to canopy conductance (Gc)
DE_Tha_Jun_2014_2 <- filter.data(DE_Tha_Jun_2014,quality.control=FALSE,
                                vars.qc=c("Tair","precip","VPD","H","LE"),
                                filter.growseas=FALSE,filter.precip=TRUE,
                                filter.vars=c("Tair","PPFD","ustar","LE"),
                                filter.vals.min=c(5,200,0.2,0),
                                filter.vals.max=c(NA,NA,NA,NA),NA.as.invalid=TRUE,
                                quality.ext="_qc",good.quality=c(0,1),
                                missing.qc.as.bad=TRUE,GPP="GPP",doy="doy",
                                year="year",tGPP=0.5,ws=15,min.int=5,precip="precip",
                                tprecip=0.1,precip.hours=24,records.per.hour=2)

# calculate Gs based on a simple gradient approach
Gs_gradient <- surface.conductance(DE_Tha_Jun_2014_2,Tair="Tair",pressure="pressure",
                                  VPD="VPD",formulation="Flux-Gradient")

summary(Gs_gradient)

# calculate Gs from the the inverted PM equation (now Rn, and Ga are needed),
# using a simple estimate of Ga based on Thom 1972
Ga <- aerodynamic.conductance(DE_Tha_Jun_2014_2,Rb_model="Thom_1972"),["Ga_h"]

# if G and/or S are available, don't forget to indicate (they are ignored by default).
# Note that Ga is not added to the data.frame 'DE_Tha_Jun_2014'
Gs_PM <- surface.conductance(DE_Tha_Jun_2014_2,Tair="Tair",pressure="pressure",
                             Rn="Rn",G="G",S=NULL,VPD="VPD",Ga=Ga,
                             formulation="Penman-Monteith")

summary(Gs_PM)

# now add Ga to the data.frame 'DE_Tha_Jun_2014' and repeat
DE_Tha_Jun_2014_2$Ga <- Ga
Gs_PM2 <- surface.conductance(DE_Tha_Jun_2014_2,Tair="Tair",pressure="pressure",
                              Rn="Rn",G="G",S=NULL,VPD="VPD",Ga="Ga",
                              formulation="Penman-Monteith")

# note the difference to the previous version (Ga="Ga")
summary(Gs_PM2)
```

umolCO2.to.gC

*Conversion between Mass and Molar Units of Carbon and CO2***Description**

Converts CO2 quantities from umol CO2 m⁻² s⁻¹ to g C m⁻² d⁻¹ and vice versa.

Usage

```
umolCO2.to.gC(CO2_flux, constants = bigleaf.constants())
```

```
gC.to.umolCO2(C_flux, constants = bigleaf.constants())
```

Arguments

CO2_flux	CO2 flux (umol CO2 m-2 s-1)
constants	Cmol - molar mass of carbon (kg mol-1) umol2mol - conversion micromole (umol) to mol (mol) mol2umol - conversion mole (mol) to micromole (umol) kg2g - conversion kilogram (kg) to gram (g) g2kg - conversion gram (g) to kilogram (kg) days2seconds - seconds per day
C_flux	Carbon (C) flux (gC m-2 d-1)

Examples

```
umolCO2.to.gC(20) # gC m-2 d-1
```

virtual.temp	<i>Virtual Temperature</i>
--------------	----------------------------

Description

Virtual temperature, defined as the temperature at which dry air would have the same density as moist air at its actual temperature.

Usage

```
virtual.temp(  
  Tair,  
  pressure,  
  VPD,  
  Esat.formula = c("Sonntag_1990", "Alduchov_1996", "Allen_1998"),  
  constants = bigleaf.constants()  
)
```

Arguments

Tair	Air temperature (deg C)
pressure	Atmospheric pressure (kPa)
VPD	Vapor pressure deficit (kPa)
Esat.formula	Optional: formula to be used for the calculation of esat and the slope of esat. One of "Sonntag_1990" (Default), "Alduchov_1996", or "Allen_1998". See Esat.slope .
constants	Kelvin - conversion degree Celsius to Kelvin eps - ratio of the molecular weight of water vapor to dry air (-)

Details

the virtual temperature is given by:

$$T_v = T_{air} / (1 - (1 - eps)e/pressure)$$

where T_{air} is in Kelvin (converted internally). Likewise, VPD is converted to actual vapor pressure (e in kPa) with `VPD.to.e` internally.

Value

T_v - virtual temperature (deg C)

References

Monteith J.L., Unsworth M.H., 2008: Principles of Environmental Physics. 3rd edition. Academic Press, London.

Examples

```
virtual.temp(25,100,1.5)
```

 VPD.to.rH

Conversions between Humidity Measures

Description

Conversion between vapor pressure (e), vapor pressure deficit (VPD), specific humidity (q), and relative humidity (rH).

Usage

```
VPD.to.rH(  
  VPD,  
  Tair,  
  Esat.formula = c("Sonntag_1990", "Alduchov_1996", "Allen_1998"),  
  constants = bigleaf.constants()  
)
```

```
rH.to.VPD(  
  rH,  
  Tair,  
  Esat.formula = c("Sonntag_1990", "Alduchov_1996", "Allen_1998"),  
  constants = bigleaf.constants()  
)
```

```
e.to.rH(  
  e,  
  Tair,  
  Esat.formula = c("Sonntag_1990", "Alduchov_1996", "Allen_1998"),  
  constants = bigleaf.constants()  
)
```

```

    e,
    Tair,
    Esat.formula = c("Sonntag_1990", "Alduchov_1996", "Allen_1998"),
    constants = bigleaf.constants()
)

VPD.to.e(
  VPD,
  Tair,
  Esat.formula = c("Sonntag_1990", "Alduchov_1996", "Allen_1998"),
  constants = bigleaf.constants()
)

e.to.VPD(
  e,
  Tair,
  Esat.formula = c("Sonntag_1990", "Alduchov_1996", "Allen_1998"),
  constants = bigleaf.constants()
)

e.to.q(e, pressure, constants = bigleaf.constants())

q.to.e(q, pressure, constants = bigleaf.constants())

q.to.VPD(
  q,
  Tair,
  pressure,
  Esat.formula = c("Sonntag_1990", "Alduchov_1996", "Allen_1998"),
  constants = bigleaf.constants()
)

VPD.to.q(
  VPD,
  Tair,
  pressure,
  Esat.formula = c("Sonntag_1990", "Alduchov_1996", "Allen_1998"),
  constants = bigleaf.constants()
)

```

Arguments

VPD	Vapor pressure deficit (kPa)
Tair	Air temperature (deg C)
Esat.formula	Optional: formula to be used for the calculation of esat and the slope of esat. One of "Sonntag_1990" (Default), "Alduchov_1996", or "Allen_1998". See Esat.slope .

constants	eps - ratio of the molecular weight of water vapor to dry air (-) Pa2kPa - conversion pascal (Pa) to kilopascal (kPa)
rH	Relative humidity (-)
e	Vapor pressure (kPa)
pressure	Atmospheric pressure (kPa)
q	Specific humidity (kg kg-1)

References

Foken, T, 2008: Micrometeorology. Springer, Berlin, Germany.

wetbulb.temp	<i>Wet-Bulb Temperature</i>
--------------	-----------------------------

Description

calculates the wet bulb temperature, i.e. the temperature that the air would have if it was saturated.

Usage

```
wetbulb.temp(
  Tair,
  pressure,
  VPD,
  accuracy = 0.001,
  Esat.formula = c("Sonntag_1990", "Alduchov_1996", "Allen_1998"),
  constants = bigleaf.constants()
)
```

Arguments

Tair	Air temperature (deg C)
pressure	Atmospheric pressure (kPa)
VPD	Vapor pressure deficit (kPa)
accuracy	Accuracy of the result (deg C)
Esat.formula	Optional: formula to be used for the calculation of esat and the slope of esat. One of "Sonntag_1990" (Default), "Alduchov_1996", or "Allen_1998". See Esat.slope .
constants	cp - specific heat of air for constant pressure (J K-1 kg-1) eps - ratio of the molecular weight of water vapor to dry air (-) Pa2kPa - conversion pascal (Pa) to kilopascal (kPa) Le067 - Lewis number for water vapor to the power of 0.67

Details

Wet-bulb temperature (T_w) is calculated from the following expression:

$$e = E_{sat}(T_w) - Le_{0.67} * \gamma * (T_{air} - T_w)$$

The equation is solved for T_w using `optimize`. Actual vapor pressure e (kPa) is calculated from VPD using the function `VPD.to.e`. The psychrometric constant γ (kPa K⁻¹) is calculated from `psychrometric.constant`. $Le_{0.67}$ is the Lewis number for water vapor to the power of 0.67 and represents the ratio of aerodynamic resistance to water vapor and heat. $Le_{0.67} * \gamma$ is sometimes referred to as the 'modified psychrometric constant (γ^*)'.

Value

T_w - wet-bulb temperature (degC)

References

Monteith J.L., Unsworth M.H., 2013: Principles of Environmental Physics. Plants, Animals, and the Atmosphere. 4th edition. Academic Press.

Examples

```
wetbulb.temp(Tair=c(20,25),pressure=100,VPD=c(1,1.6))
```

wind.profile

Wind Speed at a Given Height in the Surface Layer

Description

Wind speed at a given height above the canopy estimated from single-level measurements of wind speed.

Usage

```
wind.profile(
  data,
  z,
  Tair = "Tair",
  pressure = "pressure",
  ustar = "ustar",
  H = "H",
  wind = "wind",
  zr,
  zh,
  d = NULL,
  frac_d = 0.7,
```



```

z0m = NULL,
frac_z0m = NULL,
estimate_z0m = TRUE,
stab_correction = TRUE,
stab_formulation = c("Dyer_1970", "Businger_1971"),
constants = bigleaf.constants()
)

```

Arguments

data	Data.frame or matrix containing all required variables
z	Height above ground for which wind speed is calculated.
Tair	Air temperature (deg C)
pressure	Atmospheric pressure (kPa)
ustar	Friction velocity (m s-1)
H	Sensible heat flux (W m-2)
wind	Wind speed at height zr (m s-1); only used if stab_correction = TRUE
zr	Instrument (reference) height (m)
zh	Canopy height (m)
d	Zero-plane displacement height (-)
frac_d	Fraction of displacement height on canopy height (-); only used if d is not available
z0m	Roughness length (m), optional; only used if stab_correction = FALSE (default=0.1)
frac_z0m	Fraction of roughness length on canopy height (-), optional; only used if z0m is not provided. Default is 0.1.
estimate_z0m	Should z0m be estimated from the logarithmic wind profile? If TRUE (the default), arguments z0m and frac_z0m are ignored. See roughness.parameters for details.
stab_correction	Should stability correction be applied? Defaults to TRUE
stab_formulation	Stability correction function used (If stab_correction = TRUE). Either "Dyer_1970" or "Businger_1971".
constants	k - von-Karman constant (-) Kelvin - conversion degree Celsius to Kelvin cp - specific heat of air for constant pressure (J K-1 kg-1) g - gravitational acceleration (m s-2)

Details

The underlying assumption is the existence of a logarithmic wind profile above the height $d + z_0m$ (the height at which wind speed mathematically reaches zero according to the Monin-Obukhov similarity theory). In this case, the wind speed at a given height z is given by:

$$u(z) = (ustar/k) * (\ln((z - d)/z0m) - \psi m)$$

The roughness parameters zero-plane displacement height (d) and roughness length (z0m) can be approximated from [roughness.parameters](#). ψm is omitted if `stab_correction = FALSE` (not recommended). If `estimate_z0m = TRUE`, z0m is first estimated from the wind profile equation and then used in the equation above for the calculation of u(z) (see e.g. Newman & Klein 2014).

Value

A vector of wind speed at heights z.

Note

Note that this equation is only valid for $z \geq d + z0m$, and it is not meaningful to calculate values closely above $d + z0m$. All values in heights smaller than $d + z0m$ will return 0.

References

Monteith, J.L., Unsworth, M.H., 2008: Principles of Environmental Physics. 3rd edition. Academic Press, London.

Newman, J.F., Klein, P.M., 2014: The impacts of atmospheric stability on the accuracy of wind speed extrapolation methods. Resources 3, 81-105.

See Also

[roughness.parameters](#)

Examples

```
heights <- seq(18,40,2) # heights above ground for which to calculate wind speed
df <- data.frame(Tair=25,pressure=100,wind=c(3,4,5),ustar=c(0.5,0.6,0.65),H=c(200,230,250))
ws <- sapply(heights,function(x) wind.profile(df,z=x,zr=40,zh=25,d=16))
colnames(ws) <- paste0(heights,"m")
```

Description

Calculation of various water use efficiency (WUE) metrics.

Usage

```

WUE.metrics(
  data,
  GPP = "GPP",
  NEE = "NEE",
  LE = "LE",
  VPD = "VPD",
  Tair = "Tair",
  constants = bigleaf.constants()
)

```

Arguments

data	Data.frame or matrix containing all required variables
GPP	Gross primary productivity (umol CO2 m-2 s-1)
NEE	Net ecosystem exchange (umol CO2 m-2 s-1)
LE	Latent heat flux (W m-2)
VPD	Vapor pressure deficit (kPa)
Tair	Air temperature (deg C)
constants	Cmol - molar mass of carbon (kg mol-1) umol2mol - conversion micromole (umol) to mole (mol) kg2g - conversion kilogram (kg) to gram (g)

Details

the following metrics are calculated:

Water-use efficiency (WUE):

$$WUE = GPP/ET$$

Water-use efficiency based on NEE (WUE_NEE):

$$WUE_{NEE} = NEE/ET$$

Inherent water-use efficiency (IWUE; Beer et al. 2009):

$$IWUE = (GPP * VPD)/ET$$

Underlying water-use efficiency (uWUE; Zhou et al. 2014):

$$uWUE = (GPP * \sqrt{VPD})/ET$$

All metrics are calculated based on the median of all values. E.g. WUE = median(GPP/ET,na.rm=TRUE)

Value

a named vector with the following elements:

WUE	Water-use efficiency (gC (kg H2O)-1)
WUE_NEE	Water-use efficiency based on NEE (gC (kg H2O)-1)
IWUE	Inherent water-use efficiency (gC kPa (kg H2O)-1)
uWUE	Underlying water-use efficiency (gC kPa ^{0.5} (kg H2O)-1)

Note

Units for VPD can also be hPa. Units change accordingly. WUE_NEE is calculated based on the absolute value of NEE (the sign convention does not matter here).

References

Beer, C., et al., 2009: Temporal and among-site variability of inherent water use efficiency at the ecosystem level. *Global Biogeochemical Cycles* 23, GB2018.

Zhou, S., et al., 2014: The effect of vapor pressure deficit on water use efficiency at the sub-daily time scale. *Geophysical Research Letters* 41.

See Also

[stomatal.slope](#) for a measure of intrinsic WUE

Examples

```
## filter data for dry periods and daytime at DE-Tha in June 2014
DE_Tha_Jun_2014_2 <- filter.data(DE_Tha_Jun_2014,quality.control=FALSE,
                                vars.qc=c("Tair","precip","VPD","H","LE"),
                                filter.growseas=FALSE,filter.precip=TRUE,
                                filter.vars=c("Tair","PPFD","ustar"),
                                filter.vals.min=c(5,200,0.2),
                                filter.vals.max=c(NA,NA,NA),NA.as.invalid=TRUE,
                                quality.ext="_qc",good.quality=c(0,1),
                                missing.qc.as.bad=TRUE,GPP="GPP",doy="doy",
                                year="year",tGPP=0.5,ws=15,min.int=5,precip="precip",
                                tprecip=0.1,precip.hours=24,records.per.hour=2)

## calculate WUE metrics in the filtered periods
WUE.metrics(DE_Tha_Jun_2014_2)
```

Index

- * **datasets**
 - AT_Neu_Jul_2010, [10](#)
 - DE_Tha_Jun_2014, [18](#)
 - FR_Pue_May_2012, [30](#)
- * **humidity conversion**
 - VPD. to. rH, [85](#)
- aerodynamic.conductance, [3](#), [16](#), [23](#), [34](#), [37](#),
[38](#), [66](#), [79](#)
- air.density, [8](#)
- Arrhenius.temp.response, [9](#), [56](#)
- AT_Neu_Jul_2010, [10](#)
- bigleaf.constants, [11](#), [41](#)
- biochemical.energy, [13](#)
- D, [24](#)
- DE_Tha_Jun_2014, [18](#)
- decoupling, [14](#), [23](#)
- dew.point, [16](#)
- e.to.q (VPD.to.rH), [85](#)
- e.to.rH (VPD.to.rH), [85](#)
- e.to.VPD (VPD.to.rH), [85](#)
- energy.closure, [19](#)
- energy.use. efficiency, [21](#), [47](#)
- equilibrium.imposed.ET, [16](#), [22](#)
- Esat.slope, [15](#), [17](#), [22](#), [24](#), [57](#), [64](#), [79](#), [81](#), [84](#),
[86](#), [87](#)
- ET.to.LE (LE.to.ET), [44](#)
- extraterrestrial.radiation, [25](#)
- filter.data, [26](#)
- filter.growing.season, [29](#)
- fixInNamespace, [13](#)
- FR_Pue_May_2012, [30](#)
- Gb.Choudhury, [6](#), [7](#), [31](#), [37](#), [38](#), [66](#)
- Gb.Su, [6](#), [7](#), [34](#), [34](#), [38](#), [66](#)
- Gb.Thom, [7](#), [34](#), [37](#), [37](#), [66](#)
- gC.to.umolCO2 (umolCO2.to.gC), [83](#)
- intercellular.CO2, [38](#), [53](#), [56](#)
- isothermal.Rn, [40](#)
- kg.to.mol, [41](#)
- kinematic.viscosity, [42](#)
- latent.heat.vaporization, [43](#), [44](#), [61](#)
- LE.to.ET, [44](#)
- light.response, [45](#)
- light.use. efficiency, [21](#), [46](#)
- longwave.conductance, [47](#)
- mol.to.ms (ms.to.mol), [49](#)
- Monin.Obukhov.length, [5](#), [48](#), [72](#)
- ms.to.mol, [49](#)
- nIrob, [75](#)
- nls, [45](#), [73](#), [75](#)
- optimize, [17](#), [88](#)
- optimum.temperature, [50](#)
- photosynthetic.capacity, [52](#)
- potential.ET, [56](#)
- potential.radiation, [59](#)
- PPFD.to.Rg (Rg.to.PPFD), [65](#)
- pressure.from.elevation, [60](#), [82](#)
- psychrometric.constant, [61](#), [88](#)
- q.to.e (VPD.to.rH), [85](#)
- q.to.VPD (VPD.to.rH), [85](#)
- radiometric.surface.temp, [62](#)
- reference.ET, [63](#)
- Reynolds.Number, [64](#)
- Rg.to.PPFD, [65](#)
- rH.to.VPD (VPD.to.rH), [85](#)
- roughness.length.heat, [6](#), [66](#)
- roughness.parameters, [5](#), [66](#), [67](#), [72](#), [89](#), [90](#)
- stability.correction, [70](#)

stability.parameter, [49](#), [70](#), [71](#)
stomatal.sensitivity, [72](#)
stomatal.slope, [74](#), [92](#)
surface.CO2, [39](#), [77](#)
surface.conditions, [73](#), [75](#), [78](#)
surface.conductance, [7](#), [16](#), [23](#), [53](#), [58](#), [73](#),
[76](#), [80](#)

umolCO2.to.gC, [83](#)

virtual.temp, [84](#)
VPD.to.e, [85](#), [88](#)
VPD.to.e (VPD.to.rH), [85](#)
VPD.to.q (VPD.to.rH), [85](#)
VPD.to.rH, [85](#)

wetbulb.temp, [87](#)
wind.profile, [6](#), [35](#), [69](#), [88](#)
WUE.metrics, [90](#)