Package ‘tealeaves’

June 18, 2020

Version 1.0.5
Date 2020-06-17
Title Solve for Leaf Temperature Using Energy Balance
Depends R (>= 3.5.0), units (>= 0.6.0)
Imports checkmate (>= 2.0.0), crayon (>= 1.3.0), dplyr (>= 1.0.0),
      furrr (>= 0.1.0), future (>= 1.10.0), glue (>= 1.3.0), magrittr
      (>= 1.5.0), methods (>= 3.5.0), purrr (>= 0.3.0), rlang (>=
      0.4.0), stringr (>= 1.4.0)
Suggests testthat, knitr, rmarkdown, tidyverse, ggplot2, covr
Description Implements models of leaf temperature using energy balance. It uses units to ensure that parameters are properly specified and transformed before calculations. It allows separate lower and upper surface conductances to heat and water vapour, so sensible and latent heat loss are calculated for each surface separately as in Foster and Smith (1986) <doi:10.1111/j.1365-3040.1986.tb02108.x>. It’s straightforward to model leaf temperature over environmental gradients such as light, air temperature, humidity, and wind. It can also model leaf temperature over trait gradients such as leaf size or stomatal conductance. Other references are Monteith and Unsworth (2013, ISBN:9780123869104), Nobel (2009, ISBN:9780123741431), and Okajima et al. (2012) <doi:10.1007/s11284-011-0905-5>.
License MIT + file LICENSE
Encoding UTF-8
LazyData true
RoxygenNote 7.1.0
VignetteBuilder knitr
NeedsCompilation no
Author Chris Muir [aut, cre] (<https://orcid.org/0000-0003-2555-3878>)
Maintainer Chris Muir <cdmuir@hawaii.edu>
Repository CRAN
Date/Publication 2020-06-18 10:30:02 UTC
R topics documented:

-get_dwv .......................................................... 2
-get_Dx .............................................................. 4
-get_gbw ............................................................ 5
-get_gh .............................................................. 6
-get_gr .............................................................. 7
-get_gtw ............................................................ 8
-get_H ............................................................... 10
-get_hvap ........................................................... 11
-get_L ............................................................... 12
-get_nu ............................................................. 13
-get_Pa ............................................................. 14
-get_ps ............................................................. 15
-get_Rabs .......................................................... 16
-get_re ............................................................. 17
-get_sh ............................................................. 18
-get_Sr ............................................................. 19
-get_Tv ............................................................. 20
-Ar ................................................................. 22
-constants .......................................................... 23
-convert_conductance .......................................... 23
-E ................................................................. 24
-energy_balance .................................................. 25
-enviro_par ........................................................ 26
-leaf_par ........................................................... 27
-make_parameters ................................................. 27
-parameter_names ................................................ 29
-tealeaves .......................................................... 29
-tleaves ........................................................... 30
-tl_example1 ........................................................ 32

Index 33

---

-get_dwv d_wv: water vapour gradient (mol / m ^ 3)

Description
d_wv: water vapour gradient (mol / m ^ 3)

Usage

-get_dwv(T_leaf, pars, unitless)
Arguments

- **T_leaf**: Leaf temperature in Kelvin
- **pars**: Concatenated parameters (leaf_par, enviro_par, and constants)
- **unitless**: Logical. Should function use parameters with units? The function is faster when FALSE, but input must be in correct units or else results will be incorrect without any warning.

Details

**Water vapour gradient**: The water vapour pressure differential from inside to outside of the leaf is the saturation water vapor pressure inside the leaf ($p_{\text{leaf}}$) minus the water vapor pressure of the air ($p_{\text{air}}$):

$$d_{wv} = \frac{p_{\text{leaf}}}{RT_{\text{leaf}}} - RH \frac{p_{\text{air}}}{RT_{\text{air}}}$$

Note that water vapor pressure is converted from kPa to mol / m\(^3\) using ideal gas law.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>(R)</th>
<th>Description</th>
<th>Units</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>(p_{\text{air}})</td>
<td>(p_{\text{air}})</td>
<td>saturation water vapour pressure of air</td>
<td>kPa</td>
<td>calculated</td>
</tr>
<tr>
<td>(p_{\text{leaf}})</td>
<td>(p_{\text{leaf}})</td>
<td>saturation water vapour pressure inside the leaf</td>
<td>kPa</td>
<td>calculated</td>
</tr>
<tr>
<td>(R)</td>
<td>(R)</td>
<td>ideal gas constant</td>
<td>J / (mol K)</td>
<td>8.3144598</td>
</tr>
<tr>
<td>RH</td>
<td>RH</td>
<td>relative humidity</td>
<td>%</td>
<td>0.50</td>
</tr>
<tr>
<td>(T_{\text{air}})</td>
<td>(T_{\text{air}})</td>
<td>air temperature</td>
<td>K</td>
<td>298.15</td>
</tr>
<tr>
<td>(T_{\text{leaf}})</td>
<td>(T_{\text{leaf}})</td>
<td>leaf temperature</td>
<td>K</td>
<td>input</td>
</tr>
</tbody>
</table>

Value

Value in mol / m\(^3\) of class units

Examples

```r
# Water vapour gradient:

leaf_par <- make_leafpar()
enviro_par <- make_enviropar()
constants <- make_constants()
pars <- c(leaf_par, enviro_par, constants)
T_leaf <- set_units(300, K)
T_air <- set_units(298.15, K)
p_leaf <- set_units(35.31683, kPa)
p_air <- set_units(31.65367, kPa)

d_wv <- p_leaf / (pars$R * T_leaf) - pars$RH * p_air / (pars$R * T_air)
```
D_x: Calculate diffusion coefficient for a given temperature and pressure

**Usage**

```
.get_Dx(D_0, Temp, eT, P, unitless)
```

**Arguments**

- **D_0**: Diffusion coefficient at 273.15 K (0 °C) and 101.3246 kPa
- **Temp**: Temperature in Kelvin
- **eT**: Exponent for temperature dependence of diffusion
- **P**: Atmospheric pressure in kPa
- **unitless**: Logical. Should function use parameters with units? The function is faster when FALSE, but input must be in correct units or else results will be incorrect without any warning.

**Details**

\[
D = D_0 \left( \frac{T}{273.15} \right)^{eT} \left( \frac{101.3246}{P} \right)
\]

According to Montieth & Unger (2013), eT is generally between 1.5 and 2. Their data in Appendix 3 indicate eT = 1.75 is reasonable for environmental physics.

**Value**

Value in m²/s of class `units`

**References**


**Examples**

```
tealeaves:::get_Dx(
    D_0 = set_units(2.12e-05, m^2/s),
    Temp = set_units(298.15, K),
    eT = set_units(1.75),
    P = set_units(101.3246, kPa),
```

Description

g_bw: Boundary layer conductance to water vapour (m / s)

Usage

.get_gbw(T_leaf, surface, pars, unitless)

Arguments

T_leaf Leaf temperature in Kelvin
surface Leaf surface (lower or upper)
pars Concatenated parameters (leaf_par, enviro_par, and constants)
unitless Logical. Should function use parameters with units? The function is faster when FALSE, but input must be in correct units or else results will be incorrect without any warning.

Details

\[ g_{bw} = \frac{D_w Sh}{d} \]

<table>
<thead>
<tr>
<th>Symbol</th>
<th>R</th>
<th>Description</th>
<th>Units</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>(d)</td>
<td>leafsize</td>
<td>Leaf characteristic dimension in meters</td>
<td>m</td>
<td>0.1</td>
</tr>
<tr>
<td>(D_w)</td>
<td>(D_w)</td>
<td>diffusion coefficient for water vapour</td>
<td>m^2 / s</td>
<td>calculated</td>
</tr>
<tr>
<td>(Sh)</td>
<td>(Sh)</td>
<td>Sherwood number</td>
<td>none</td>
<td>calculated</td>
</tr>
</tbody>
</table>

Value

Value in m / s of class units

Examples

```
library(tealeaves)

cs <- make_constants()
ep <- make_enviropar()
lp <- make_leafpar()
```
T_leaf <- set_units(298.15, K)

tealeaves:::.get_gbwr(T_leaf, "lower", c(cs, ep, lp), FALSE)

---

**Description**

\(g_h\): boundary layer conductance to heat (m/s)

**Usage**

```
.get_gh(T_leaf, surface, pars, unitless)
```

**Arguments**

- `T_leaf`: Leaf temperature in Kelvin
- `surface`: Leaf surface (lower or upper)
- `pars`: Concatenated parameters (`leaf_par`, `enviro_par`, and `constants`)
- `unitless`: Logical. Should function use parameters with units? The function is faster when FALSE, but input must be in correct units or else results will be incorrect without any warning.

**Details**

\[
g_h = \frac{D_h \cdot Nu}{d}
\]

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
<th>Units</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>(d)</td>
<td>Leaf characteristic dimension in meters</td>
<td>m</td>
<td>0.1</td>
</tr>
<tr>
<td>(D_h)</td>
<td>Diffusion coefficient for heat in air</td>
<td>m²/s</td>
<td>Calculated</td>
</tr>
<tr>
<td>(Nu)</td>
<td>Nusselt number</td>
<td>none</td>
<td>Calculated</td>
</tr>
</tbody>
</table>

**Value**

Value in m/s of class `units`

**Examples**

```r
library(tealeaves)

cs <- make_constants()
ep <- make_enviropar()
lp <- make_leafpar()
```
\texttt{T\_leaf <- set\_units(298.15, K)}

\texttt{tealeaves:::get\_gh(T\_leaf, "lower", c(cs, ep, lp), FALSE)}

---

**get\_gr**

---

**Gr: Grashof number**

**Description**

Gr: Grashof number

**Usage**

\texttt{.get\_gr(T\_leaf, pars, unitless)}

**Arguments**

- \texttt{T\_leaf}: Leaf temperature in Kelvin
- \texttt{pars}: Concatenated parameters (leaf\_par, enviro\_par, and constants)
- \texttt{unitless}: Logical. Should function use parameters with units? The function is faster when FALSE, but input must be in correct units or else results will be incorrect without any warning.

**Details**

\[ Gr = t\_air G d^3 |T\_v,\text{leaf} - T\_v,\text{air}| / D_m^2 \]

<table>
<thead>
<tr>
<th>Symbol</th>
<th>R</th>
<th>Description</th>
<th>Units</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>(d)</td>
<td>(l)</td>
<td>leaf characteristic dimension in meters</td>
<td>m</td>
<td>0.1</td>
</tr>
<tr>
<td>(D_m)</td>
<td>(D_m)</td>
<td>diffusion coefficient of momentum in air</td>
<td>m(^2)/s</td>
<td>calculated</td>
</tr>
<tr>
<td>(G)</td>
<td>(G)</td>
<td>gravitational acceleration</td>
<td>m/s(^2)</td>
<td>9.8</td>
</tr>
<tr>
<td>(t_air)</td>
<td>(t_air)</td>
<td>coefficient of thermal expansion of air</td>
<td>1/K</td>
<td>1/Tmp</td>
</tr>
<tr>
<td>(T_v,\text{air})</td>
<td>(T_v,\text{air})</td>
<td>virtual air temperature</td>
<td>K</td>
<td>calculated</td>
</tr>
<tr>
<td>(T_v,\text{leaf})</td>
<td>(T_v,\text{leaf})</td>
<td>virtual leaf temperature</td>
<td>K</td>
<td>calculated</td>
</tr>
</tbody>
</table>

**Value**

A unitless number of class \texttt{units}

**Examples**

\texttt{library(tealeaves)}
cs <- make_constants()
ep <- make_enviropar()
lp <- make_leafpar()

T_leaf <- set_units(298.15, K)
teaLeaves:::.get_gr(T_leaf, c(cs, ep, lp), FALSE)

---

**get_gtw**

*Description*

g_tw: total conductance to water vapour (m/s)

**Usage**

.get_gtw(T_leaf, pars, unitless)

**Arguments**

- **T_leaf**: Leaf temperature in Kelvin
- **pars**: Concatenated parameters (leaf_par, enviro_par, and constants)
- **unitless**: Logical. Should function use parameters with units? The function is faster when FALSE, but input must be in correct units or else results will be incorrect without any warning.

**Details**

**Total conductance to water vapor**: The total conductance to water vapor (g_tw) is the sum of the parallel lower (abaxial) and upper (adaxial) conductances:

\[
g_{tw} = g_{w, lower} + g_{w, upper}
\]

The conductance to water vapor on each surface is a function of parallel stomatal (g_{sw}) and cuticular (g_{uw}) conductances in series with the boundary layer conductance (g_{bw}). The stomatal, cuticular, and boundary layer conductance on the lower surface are:

\[
g_{sw, lower} = g_{sw}(1 - sr)R(T_{leaf} + T_{air})/2  
g_{uw, lower} = g_{uw}/2R(T_{leaf} + T_{air})/2
\]

See **get_gbw** for details on calculating boundary layer conductance. The equations for the upper surface are:

\[
g_{sw, upper} = g_{sw}srR(T_{leaf} + T_{air})/2
\]
Note that the stomatal and cuticular conductances are given in units of $(\mu\text{mol H}_2\text{O})/(\text{m}^2 \text{ s Pa})$ (see `make_leafpar`) and converted to m/s using the ideal gas law. The total leaf stomatal ($g_{sw}$) and cuticular ($g_{uw}$) conductances are partitioned across lower and upper surfaces. The stomatal conductance on each surface depends on stomatal ratio ($sr$); the cuticular conductance is assumed identical on both surfaces.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
<th>Units</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>$g_{sw}$</td>
<td>stomatal conductance to H$_2$O</td>
<td>$(\mu\text{mol H}_2\text{O})/(\text{m}^2 \text{ s Pa})$</td>
<td>5</td>
</tr>
<tr>
<td>$g_{uw}$</td>
<td>cuticular conductance to H$_2$O</td>
<td>$(\mu\text{mol H}_2\text{O})/(\text{m}^2 \text{ s Pa})$</td>
<td>0.1</td>
</tr>
<tr>
<td>$R$</td>
<td>ideal gas constant</td>
<td>J/(mol K)</td>
<td>8.3144598</td>
</tr>
<tr>
<td>logit($sr$)</td>
<td>stomatal ratio (logit transformed)</td>
<td>none</td>
<td>0 = logit(0.5)</td>
</tr>
<tr>
<td>$T_{air}$</td>
<td>air temperature</td>
<td>K</td>
<td>298.15</td>
</tr>
<tr>
<td>$T_{leaf}$</td>
<td>leaf temperature</td>
<td>K</td>
<td>input</td>
</tr>
</tbody>
</table>

**Value**

Value in m/s of class units

**Examples**

```r
## Total conductance to water vapor

# Hypostomatous leaf; default parameters
leaf_par <- make_leafpar(replace = list(logit_sr = set_units(-Inf)))
enviro_par <- make_enviropar()
constants <- make_constants()
pars <- c(leaf_par, enviro_par, constants)
T_leaf <- set_units(300, K)

## Fixing boundary layer conductance rather than calculating
gbw_lower <- set_units(0.1, m / s)
gbw_upper <- set_units(0.1, m / s)

# Lower surface ----
## Note that pars$logit_sr is logit-transformed! Use stats::plogis() to convert to proportion.
gsw_lower <- set_units(pars$g_sw * (set_units(1) - stats::plogis(pars$logit_sr)) * pars$R * ((T_leaf + pars$T_air) / 2), "m / s")
guw_lower <- set_units(pars$g_uw * 0.5 * pars$R * ((T_leaf + pars$T_air) / 2), m / s)
gtw_lower <- 1 / (1 / (gsw_lower + guw_lower) + 1 / gbw_lower)

# Upper surface ----
gsw_upper <- set_units(pars$g_sw * stats::plogis(pars$logit_sr) * pars$R * ((T_leaf + pars$T_air) / 2), m / s)
guw_upper <- set_units(pars$g_uw * 0.5 * pars$R * ((T_leaf + pars$T_air) / 2), m / s)
gtw_upper <- 1 / (1 / (gsw_upper + guw_upper) + 1 / gbw_upper)

## Lower and upper surface are in parallel

g_tw <- gtw_lower + gtw_upper
```
Description

H: sensible heat flux density (W / m^2)

Usage

.get_H(T_leaf, pars, unitless)

Arguments

T_leaf Leaf temperature in Kelvin
pars Concatenated parameters (leaf_par, enviro_par, and constants)
unitless Logical. Should function use parameters with units? The function is faster when FALSE, but input must be in correct units or else results will be incorrect without any warning.

Details

\[ H = P_a c_p g_h (T_{leaf} - T_{air}) \]

<table>
<thead>
<tr>
<th>Symbol</th>
<th>R</th>
<th>Description</th>
<th>Units</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>c_p</td>
<td>c_p</td>
<td>heat capacity of air</td>
<td>J / (g K)</td>
<td>1.01</td>
</tr>
<tr>
<td>g_h</td>
<td>g_h</td>
<td>boundary layer conductance to heat</td>
<td>m / s</td>
<td>calculated</td>
</tr>
<tr>
<td>P_a</td>
<td>P_a</td>
<td>density of dry air</td>
<td>g / m^3</td>
<td>calculated</td>
</tr>
<tr>
<td>T_air</td>
<td>T_air</td>
<td>air temperature</td>
<td>K</td>
<td>298.15</td>
</tr>
<tr>
<td>T_leaf</td>
<td>T_leaf</td>
<td>leaf temperature</td>
<td>K</td>
<td>input</td>
</tr>
</tbody>
</table>

Value

Value in W / m^2 of class units

See Also

.get_gh, .get_Pa

Examples

library(tealeaves)

cs <- make_constants()
ep <- make_enviropar()
lp <- make_leafpar()

T_leaf <- set_units(298.15, K)
tealeaves::.get_H(T_leaf, c(cs, ep, lp), FALSE)

Description

h_vap: heat of vaporization (J / mol)

Usage

.get_hvap(T_leaf, unitless)

Arguments

T_leaf Leaf temperature in Kelvin
unitless Logical. Should function use parameters with units? The function is faster when FALSE, but input must be in correct units or else results will be incorrect without any warning.

Details

Heat of vaporization: The heat of vaporization (h_vap) is a function of temperature. I used data from on temperature and h_vap from Nobel (2009, Appendix 1) to estimate a linear regression. See Examples.

Value

Value in J/mol of class units

References


Examples

# Heat of vaporization and temperature
## data from Nobel (2009)
T_K <- 273.15 + c(0, 10, 20, 25, 30, 40, 50, 60)
h_vap <- 1e3 * c(45.06, 44.63, 44.21, 44.00,
fit <- lm(h_vap ~ T_K)  # (in J / mol)
coef(fit)

T_leaf <- 298.15
h_vap <- set_units(56847.68250, J / mol) - set_units(43.12514, J / mol / K) * set_units(T_leaf, K)

## h_vap at 298.15 K is 43989.92 [J/mol]
set_units(h_vap, J / mol)
tealeaves:::.get_hvap(set_units(298.15, K), FALSE)

---

**Description**

L: Latent heat flux density (W / m^2)

**Usage**

.get_L(T_leaf, pars, unitless)

**Arguments**

- **T_leaf**: Leaf temperature in Kelvin
- **pars**: Concatenated parameters (leaf_par, enviro_par, and constants)
- **unitless**: Logical. Should function use parameters with units? The function is faster when FALSE, but input must be in correct units or else results will be incorrect without any warning.

**Details**

\[ L = h_{vap} g_{tw} d_{wv} \]

<table>
<thead>
<tr>
<th>Symbol</th>
<th>R</th>
<th>Description</th>
<th>Units</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>d_{wv}</td>
<td>d_wv</td>
<td>water vapour gradient</td>
<td>mol / m ^ 3</td>
<td>calculated</td>
</tr>
<tr>
<td>h_{vap}</td>
<td>h_vap</td>
<td>latent heat of vaporization</td>
<td>J / mol</td>
<td>calculated</td>
</tr>
<tr>
<td>g_{tw}</td>
<td>g_tw</td>
<td>total conductance to H2O</td>
<td>(μmol H2O) / (m^2 s Pa)</td>
<td>calculated</td>
</tr>
</tbody>
</table>
.get_nu

Value
Value in W / m\(^2\) of class units

Examples

library(tealeaves)
cs <- make_constants()
ep <- make_enviropar()
lp <- make_leafpar()
T_leaf <- set_units(298.15, K)
tealeaves:::.get_L(T_leaf, c(cs, ep, lp), FALSE)

---

.get_nu Nu: Nusselt number

Description

Nu: Nusselt number

Usage

.get_nu(T_leaf, surface, pars, unitless)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>T_leaf</td>
<td>Leaf temperature in Kelvin</td>
</tr>
<tr>
<td>surface</td>
<td>Leaf surface (lower or upper)</td>
</tr>
<tr>
<td>pars</td>
<td>Concatenated parameters (leaf_par, enviropar, and constants)</td>
</tr>
<tr>
<td>unitless</td>
<td>Logical. Should function use parameters with units? The function is faster when FALSE, but input must be in correct units or else results will be incorrect without any warning.</td>
</tr>
</tbody>
</table>

Details

The Nusselt number depends on a combination how much free or forced convection predominates. For mixed convection:

\[ Nu = (aRe^b)^{3.5} + (eGr^d)^{3.5} \]
### Description

$P_a$: density of dry air (g/m³)

### Usage

```
.get_Pa(T_leaf, pars, unitless)
```

### Arguments

- **T_leaf**: Leaf temperature in Kelvin
- **pars**: Concatenated parameters (leaf_par, enviro_par, and constants)
- **unitless**: Logical. Should function use parameters with units? The function is faster when FALSE, but input must be in correct units or else results will be incorrect without any warning.

### Details

\[
P_a = \frac{P}{R_{\text{air}}(T_{\text{leaf}} - T_{\text{air}})/2}
\]
### Description

Saturation water vapour pressure (kPa)

### Usage

`get_ps(Temp, P, unitless)`

### Arguments

- **Temp**: Temperature in Kelvin
- **P**: Atmospheric pressure in kPa
- **unitless**: Logical. Should function use parameters with units? The function is faster when FALSE, but input must be in correct units or else results will be incorrect without any warning.

### Details

Goff-Gratch equation (see http://cires1.colorado.edu/~voemel/vp.html)

This equation assumes $P = 1$ atm = 101.3246 kPa, otherwise boiling temperature needs to change.
Value

Value in kPa of class units

References

http://cires1.colorado.edu/~voemel/vp.html

Examples

```r
T_leaf <- set_units(298.15, K)
P <- set_units(101.3246, kPa)
tealeaves:::.get_ps(T_leaf, P, FALSE)
```

---

### .get_Rabs

**Description**

R_abs: total absorbed radiation (W / m\(^2\))

**Usage**

`.get_Rabs(pars, unitless)`

**Arguments**

- **pars**: Concatenated parameters (`leaf_par`, `enviro_par`, and `constants`)
- **unitless**: Logical. Should function use parameters with units? The function is faster when FALSE, but input must be in correct units or else results will be incorrect without any warning.

**Details**

The following treatment follows Okajima et al. (2012):

\[
R_{\text{abs}} = \alpha_s (1 + r) S_{\text{sw}} + \alpha_l \sigma (T_{\text{sky}}^4 + T_{\text{air}}^4)
\]

The incident longwave (aka thermal infrared) radiation is modeled from sky and air temperature \(\sigma(T_{\text{sky}}^4 + T_{\text{air}}^4)\) where \(T_{\text{sky}}\) is function of the air temperature and incoming solar shortwave radiation:

\[
T_{\text{sky}} = T_{\text{air}} - 20S_{\text{sw}}/1000
\]

<table>
<thead>
<tr>
<th>Symbol</th>
<th>R</th>
<th>Description</th>
<th>Units</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\alpha_s)</td>
<td>abs_s</td>
<td>absorbtivity of shortwave radiation (0.3 - 4 (\mu)m)</td>
<td>none</td>
<td>0.80</td>
</tr>
</tbody>
</table>
### .get_re

*Re: Reynolds number*

#### Description

Re: Reynolds number

#### Usage

```
.get_re(T_leaf, pars, unitless)
```

#### Arguments

- **T_leaf**: Leaf temperature in Kelvin
- **pars**: Concatenated parameters (`leaf_par`, `enviro_par`, and `constants`)
- **unitless**: Logical. Should function use parameters with units? The function is faster when `FALSE`, but input must be in correct units or else results will be incorrect without any warning.

### Value

Value in $W / m^2$ of class units

### References


### Examples

```r
library(tealeaves)

cs <- make_constants()
ep <- make_enviropar()
lp <- make_leafpar()
ep$T_sky <- ep$T_sky(ep)

.tealeaves:::.get_Rabs(c(cs, ep, lp), FALSE)
```

---

### Table

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\alpha_l$</td>
<td>absorbptivity of longwave radiation (4 - 80 $\mu$m)</td>
</tr>
<tr>
<td>$r$</td>
<td>reflectance for shortwave irradiance (albedo)</td>
</tr>
<tr>
<td>$\sigma$</td>
<td>Stefan-Boltzmann constant</td>
</tr>
<tr>
<td>$S_{sw}$</td>
<td>incident short-wave (solar) radiation flux density</td>
</tr>
<tr>
<td>$S_{lw}$</td>
<td>incident long-wave radiation flux density</td>
</tr>
<tr>
<td>$T_{air}$</td>
<td>air temperature</td>
</tr>
<tr>
<td>$T_{sky}$</td>
<td>sky temperature</td>
</tr>
</tbody>
</table>

### Value

- $\alpha_l$: none, 0.97
- $r$: none, 0.2
- $\sigma$: $W / (m^2 K^4)$, 5.67e-08
- $S_{sw}$: $W / m^2$, 1000
- $S_{lw}$: $W / m^2$, calculated
- $T_{air}$: $K$, 298.15
- $T_{sky}$: $K$, calculated
Details

\[ Re = \frac{ud}{D_m} \]

<table>
<thead>
<tr>
<th>Symbol</th>
<th>( R )</th>
<th>Description</th>
<th>Units</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>( d )</td>
<td>leafsize</td>
<td>Leaf characteristic dimension in meters</td>
<td>m</td>
<td>0.1</td>
</tr>
<tr>
<td>( D_m )</td>
<td>( D_m )</td>
<td>diffusion coefficient of momentum in air</td>
<td>m²/s</td>
<td>calculated</td>
</tr>
<tr>
<td>( u )</td>
<td>wind</td>
<td>windspeed</td>
<td>m/s</td>
<td>2</td>
</tr>
</tbody>
</table>

Value

A unitless number of class units

Examples

library(tealeaves)

cs <- make_constants()
ep <- make_enviropar()
lp <- make_leafpar()

T_leaf <- set_units(298.15, K)

tealeaves:::.get_re(T_leaf, c(cs, ep, lp), FALSE)

Sh: Sherwood number

Description

Sh: Sherwood number

Usage

.get_sh(T_leaf, surface, pars, unitless)

Arguments

- \( T_{\text{leaf}} \): Leaf temperature in Kelvin
- surface: Leaf surface (lower or upper)
- pars: Concatenated parameters (leaf_par, enviro_par, and constants)
- unitless: Logical. Should function use parameters with units? The function is faster when FALSE, but input must be in correct units or else results will be incorrect without any warning.
Details

The Sherwood number depends on a combination how much free or forced convection predominates. For mixed convection:

\[
Sh = (aRe^b)^{3.5} + (cGr^d)^{3.5})^{1/3.5}
\]

<table>
<thead>
<tr>
<th>Symbol</th>
<th>R</th>
<th>Description</th>
<th>Units</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>a, b, c, d</td>
<td>a, b, c, d</td>
<td>empirical coefficients</td>
<td>none</td>
<td>calculated</td>
</tr>
<tr>
<td>Gr</td>
<td>Gr</td>
<td>Grashof number</td>
<td>none</td>
<td>calculated</td>
</tr>
<tr>
<td>Re</td>
<td>Re</td>
<td>Reynolds number</td>
<td>none</td>
<td>calculated</td>
</tr>
</tbody>
</table>

Value

A unitless number of class units

Examples

```r
library(tealeaves)

cs <- make_constants()
ep <- make_enviropar()
lp <- make_leafpar()

T_leaf <- set_units(298.15, K)

tea1eaves:::.get_sh(T_leaf, "lower", c(cs, ep, lp), FALSE)
```

Description

S_r: longwave re-radiation (W / m^2)

Usage

`.get_Sr(T_leaf, pars)`

Arguments

- **T_leaf** Leaf temperature in Kelvin
- **pars** Concatenated parameters (leaf_par, enviropar, and constants)
\[ S_r = 2\sigma \alpha_l T_{\text{air}}^4 \]

The factor of 2 accounts for re-radiation from both leaf surfaces (Foster and Smith 1986).

<table>
<thead>
<tr>
<th>Symbol</th>
<th>( R )</th>
<th>Description</th>
<th>Units</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \alpha_l )</td>
<td>( \text{abs}_l )</td>
<td>absorbtivity of longwave radiation (4 - 80 ( \mu )m)</td>
<td>none</td>
<td>0.97</td>
</tr>
<tr>
<td>( T_{\text{air}} )</td>
<td>( T_{\text{air}} )</td>
<td>air temperature</td>
<td>K</td>
<td>298.15</td>
</tr>
<tr>
<td>( \sigma )</td>
<td>( \text{s} )</td>
<td>Stefan-Boltzmann constant</td>
<td>W / (m(^2) K(^4))</td>
<td>5.67e-08</td>
</tr>
</tbody>
</table>

Note that leaf absorbtivity is the same value as leaf emissivity.

**Value**

Value in W / m\(^2\) of class units

**References**


**Examples**

```r
library(tealeaves)

cs <- make_constants()
ep <- make_enviropar()
lp <- make_leafpar()

T_leaf <- set_units(298.15, K)

tealeaves:::.get_Sr(T_leaf, c(cs, ep, lp))
```

---

**.get_Tv**

*Calculate virtual temperature*

**Description**

Calculate virtual temperature

**Usage**

```
.get_Tv(Temp, p, P, epsilon, unitless)
```
.get_Tv

Arguments

<table>
<thead>
<tr>
<th>Symbol</th>
<th>R</th>
<th>Description</th>
<th>Units</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>Temp</td>
<td></td>
<td>Temperature in Kelvin</td>
<td></td>
<td></td>
</tr>
<tr>
<td>p</td>
<td>p</td>
<td>water vapour pressure in kPa</td>
<td>kPa</td>
<td>calculated</td>
</tr>
<tr>
<td>P</td>
<td>P</td>
<td>Atmospheric pressure in kPa</td>
<td>kPa</td>
<td>101.3246</td>
</tr>
<tr>
<td>epsilon</td>
<td>epsilon</td>
<td>ratio of water to air molar masses (unitless)</td>
<td>unitless</td>
<td>0.622</td>
</tr>
<tr>
<td>unitless</td>
<td></td>
<td>Logical. Should function use parameters with units? The function is faster when FALSE, but input must be in correct units or else results will be incorrect without any warning.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Details

\[ T_V = T / \left[ 1 - (1 - \epsilon)(p/P) \right] \]

Eq. 2.35 in Monteith & Unsworth (2013)

Value

Value in K of class units

References


Examples

```r
library(tealeaves)
cs <- make_constants()
ep <- make_enviropar()
lp <- make_leafpar()

T_leaf <- set_units(298.15, K)
p <- ep$RH * tealeaves:::.get_ps(T_leaf, ep$P, FALSE)
tealeaves:::.get_Tv(T_leaf, p, ep$P, cs$epsilon, FALSE)
```
Description

$Ar$: Archimedes number

Usage

$Ar(T_{\text{leaf}}, \text{pars}, \text{unitless} = \text{FALSE})$

Arguments

- **$T_{\text{leaf}}$**: Leaf temperature in Kelvin
- **pars**: Concatenated parameters (`leaf_par`, `enviro_par`, and `constants`)
- **unitless**: Logical. Should function use parameters with units? The function is faster when `FALSE`, but input must be in correct units or else results will be incorrect without any warning.

Details

The Archimedes number is a dimensionless number that describes when free or forced convection dominates.

$$Ar = \frac{Gr}{Re^2}$$

<table>
<thead>
<tr>
<th>Symbol</th>
<th>R</th>
<th>Description</th>
<th>Units</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>$Gr$</td>
<td>Gr</td>
<td>Grashof number</td>
<td>none</td>
<td>calculated</td>
</tr>
<tr>
<td>$Re$</td>
<td>Re</td>
<td>Reynolds number</td>
<td>none</td>
<td>calculated</td>
</tr>
</tbody>
</table>

Value

- `unitless = TRUE`: A unitless number of class `numeric`
- `unitless = FALSE`: A unitless number of class `units` Also returns Reynolds and Grashof numbers

Examples

```r
cs <- make_constants()
ep <- make_enviropar()
lp <- make_leafpar()
pars <- c(cs, lp, ep)
T_leaf <- set_units(298.15, "K")
Ar(T_leaf, pars)
```
**constants**

*S3 class constants*

**Description**

Constructor function for constants class. This function ensures that physical constant inputs are properly formatted.

**Usage**

\[
\text{constants(.x)}
\]

**Arguments**

- **.x** A list to be constructed into **constants**. If units are not provided, they will be set without conversion. If units are provided, they will be checked and converted to units that **tealeaves** uses.

**convert_conductance**

*Convert conductance units*

**Description**

Convert conductance units

**Usage**

\[
\text{convert_conductance(.g, Temp = NULL, P = NULL)}
\]

**Arguments**

- **.g** Conductance in class units. Units must convertible to one of "m/s", "umol/m^2/s/Pa", or "mol/m^2/s"
- **Temp** A temperature value of class units
- **P** A pressure value of class units that is convertible to kPa

**Value**

A list of three values of class units with units "m/s", "umol/m^2/s/Pa", and "mol/m^2/s".
Examples

```r
g_sw <- set_units(10, "m/s")
convert_conductance(g_sw,
  Temp = set_units(298.15, "K"),
  P = set_units(101.3246, "kPa"))

g_sw <- set_units(4, "umol/m^2/s/Pa")
convert_conductance(g_sw,
  Temp = set_units(298.15, "K"),
  P = set_units(101.3246, "kPa"))

g_sw <- set_units(0.4, "mol/m^2/s")
convert_conductance(g_sw,
  Temp = set_units(298.15, "K"),
  P = set_units(101.3246, "kPa"))
```

Description

Evaporation (mol / (m^2 s))

Usage

```r
E(T_leaf, pars, unitless)
```

Arguments

- `T_leaf` Leaf temperature in Kelvin
- `pars` Concatenated parameters (leaf_par, enviro_par, and constants)
- `unitless` Logical. Should function use parameters with units? The function is faster when FALSE, but input must be in correct units or else results will be incorrect without any warning.

Details

The leaf evaporation rate is the product of the total conductance to water vapour (m / s) and the water vapour gradient (mol / m^3):

```
E = g_{tw} D_{wv}
```

If unitless = TRUE, T_leaf is assumed in degrees K without checking.
energy_balance

Value

unitless = TRUE: A value in units of mol / (m ^ 2 / s) number of class numeric
unitless = FALSE: A value in units of mol / (m ^ 2 / s) of class units

Examples

library(tealeaves)

cs <- make_constants()
ep <- make_enviropar()
lp <- make_leafpar()

T_leaf <- set_units(298.15, K)

E(T_leaf, c(cs, ep, lp), FALSE)

energy_balance Calculate leaf energy balance

Description

Calculate leaf energy balance

Usage

energy_balance(
  tleaf,
  leaf_par,
  enviro_par,
  constants,
  quiet = FALSE,
  components = FALSE,
  set_units = FALSE
)

Arguments

tleaf Leaf temperature in Kelvin. If input is numeric, it will be automatically con-
verted to units.

leaf_par A list of leaf parameters. This can be generated using the make_leafpar func-
tion.

enviro_par A list of environmental parameters. This can be generated using the make_enviropar
function.

constants A list of physical constants. This can be generated using the make_constants
function.
quiet Logical. Should a message appear about conversion from numeric to units? Useful for finding leaf temperature that balances heat transfer using `uniroot`.

components Logical. Should leaf energy components be returned? Transpiration (in mol / (m^2 s)) also returned.

set_units Logical. Should units be set? The function is faster when FALSE, but input must be in correct units or else results will be incorrect without any warning.

Value
A numeric value in W / m^2. Optionally, a named list of energy balance components in W / m^2 and transpiration in mol / (m^2 s).

Examples

```r
library(tealeaves)

cs <- make_constants()
ep <- make_enviropar()
lp <- make_leafpar()
ep$T_sky <- ep$T_sky(ep)

T_leaf <- set_units(298.15, K)

energy_balance(T_leaf, lp, ep, cs, FALSE, TRUE, TRUE)
```

description

Constructor function for enviro_par class. This function ensures that environmental parameter inputs are properly formatted.

Usage

`enviro_par(.x)`

Arguments

.x A list to be constructed into `enviro_par`. If units are not provided, they will be set without conversion. If units are provided, they will be checked and converted to units that tealeaves uses.
**Description**

Constructor function for leaf_par class. This function ensures that leaf parameter inputs are properly formatted.

**Usage**

```r
leaf_par(.x)
```

**Arguments**

`.x` A list to be constructed into `leaf_par`. If units are not provided, they will be set without conversion. If units are provided, they will be checked and converted to units that tealeaves uses.

---

**Description**

Make lists of parameters of leaf, environmental, or constant parameters

`make_leafpar`

`make_enviropar`

`make_constants`

**Usage**

```r
make_leafpar(replace = NULL)
```

```r
make_enviropar(replace = NULL)
```

```r
make_constants(replace = NULL)
```

**Arguments**

`replace` A named list of parameters to replace defaults. If NULL, defaults will be used.

**Details**

Leaf parameters:
### Environmental parameters:

<table>
<thead>
<tr>
<th>Symbol</th>
<th>R</th>
<th>Description</th>
<th>Units</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>d</td>
<td>d</td>
<td>Leaf characteristic dimension</td>
<td>m</td>
<td>0.1</td>
</tr>
<tr>
<td>( \alpha_l )</td>
<td>abs_l</td>
<td>absorptivity of longwave radiation (4 - 80 ( \mu )m)</td>
<td>none</td>
<td>0.97</td>
</tr>
<tr>
<td>( \alpha_s )</td>
<td>abs_s</td>
<td>absorptivity of shortwave radiation (0.3 - 4 ( \mu )m)</td>
<td>none</td>
<td>0.50</td>
</tr>
<tr>
<td>( g_{sw} )</td>
<td>g_sw</td>
<td>stomatal conductance to H2O</td>
<td>((\mu \text{mol H}_2\text{O}) / (\text{m}^2 \cdot \text{s} \cdot \text{Pa}))</td>
<td>5</td>
</tr>
<tr>
<td>( g_{uw} )</td>
<td>g_uw</td>
<td>cuticular conductance to H2O</td>
<td>((\mu \text{mol H}_2\text{O}) / (\text{m}^2 \cdot \text{s} \cdot \text{Pa}))</td>
<td>0.1</td>
</tr>
<tr>
<td>( \logit(sr) )</td>
<td>logit_sr</td>
<td>stomatal ratio (logit transformed)</td>
<td>none</td>
<td>0 = logit(0.5)</td>
</tr>
</tbody>
</table>

### Constants:

<table>
<thead>
<tr>
<th>Symbol</th>
<th>R</th>
<th>Description</th>
<th>Units</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>( P )</td>
<td>P</td>
<td>atmospheric pressure</td>
<td>kPa</td>
<td>101.3246</td>
</tr>
<tr>
<td>( r )</td>
<td>r</td>
<td>reflectance for shortwave irradiance (albedo)</td>
<td>none</td>
<td>0.2</td>
</tr>
<tr>
<td>RH</td>
<td>RH</td>
<td>relative humidity</td>
<td>none</td>
<td>0.50</td>
</tr>
<tr>
<td>( S_{sw} )</td>
<td>S_sw</td>
<td>incident short-wave (solar) radiation flux density</td>
<td>W / m(^2)</td>
<td>1000</td>
</tr>
<tr>
<td>( S_{lw} )</td>
<td>S_lw</td>
<td>incident long-wave radiation flux density</td>
<td>W / m(^2)</td>
<td>Calculated</td>
</tr>
<tr>
<td>( T_{air} )</td>
<td>T_air</td>
<td>air temperature</td>
<td>K</td>
<td>298.15</td>
</tr>
<tr>
<td>( u )</td>
<td>wind</td>
<td>windspeed</td>
<td>m / s</td>
<td>2</td>
</tr>
</tbody>
</table>

### Value

- `make_leafpar`: An object inheriting from class `leaf_par`
- `make_enviropar`: An object inheriting from class `enviro_par`
- `make_constants`: An object inheriting from class `constants`

### Examples

```r
library(tealeaves)
# Use defaults
cs <- make_constants()
ep <- make_enviropar()
```
lp <- make_leafpar()

# Replace defaults

ep <- make_enviropar(
  replace = list(
    T_air = set_units(300, K)
  )
)

lp <- make_leafpar(
  replace = list(
    leafsize = set_units(c(0.1, 0.2), m)
  )
)

---

**parameter_names**

*Get vector of parameter names*

**Description**

Get vector of parameter names

**Usage**

`parameter_names(which)`

**Arguments**

*which* A character string indicating which parameter names to retrieve, "constants", "enviro", or "leaf". Partial matching allowed.

**Examples**

`parameter_names("leaf")`

---

**tealeaves**

*tealeaves package*

**Description**

Solve for Leaf Temperature Using Energy Balance

**Details**

See the README on GitHub
Description

tleaves: find leaf temperatures for multiple parameter sets
tleaf: find leaf temperatures for a single parameter set

Usage

tleaves(
  leaf_par,
  enviro_par,
  constants,
  progress = TRUE,
  quiet = FALSE,
  set_units = TRUE,
  parallel = FALSE
)

tleaf(leaf_par, enviro_par, constants, quiet = FALSE, set_units = TRUE)

Arguments

leaf_par        A list of leaf parameters. This can be generated using the make_leafpar function.
enviro_par      A list of environmental parameters. This can be generated using the make_enviropar function.
constants       A list of physical constants. This can be generated using the make_constants function.
progress        Logical. Should a progress bar be displayed?
quiet           Logical. Should messages be displayed?
set_units       Logical. Should units be set? The function is faster when FALSE, but input must be in correct units or else results will be incorrect without any warning.
parallel        Logical. Should parallel processing be used via future_map?

Value

tleaves:

A tibble with the following units columns

Input:

abs_l          Absorbtivity of longwave radiation (unitless)
Absorptivity of shortwave radiation (unitless)
Stomatal conductance to H2O (µmol H2O / (m² s Pa))
Cuticular conductance to H2O (µmol H2O / (m² s Pa))
(m)
Stomatal ratio (logit transformed; unitless)
Atmospheric pressure (kPa)
Relative humidity (unitless)
incident long-wave radiation flux density (W / m²)
incident short-wave (solar) radiation flux density (W / m²)
Air temperature (K)
Wind speed (m / s)
Equilibrium leaf temperature (K)
Leaf energy balance (W / m²) at tleave
Convergence code (0 = converged)
Total absorbed radiation (W / m²; see .get_Rabs)
Thermal infrared radiation loss (W / m²; see .get_Sr)
Sensible heat flux density (W / m²; see .get_H)
Latent heat flux density (W / m²; see .get_L)
Evapotranspiration (mol H2O / (m² s))

Examples

# tleaf for single parameter set:
leaf_par <- make_leafpar()
enviro_par <- make_enviropar()
constants <- make_constants()
tleaf(leaf_par, enviro_par, constants)

# tleaves for multiple parameter set:
enviro_par <- make_enviropar(
  replace = list(T_air = set_units(c(293.15, 298.15), K)))
Describing 32

\text{tl_example1}

\text{tleaves(leaf_par, enviro_par, constants)}

---

\text{tl_example1} \quad \text{tealeaves example output 1}

---

**Description**

An example output from the \text{tealeaves} function.

**Usage**

\text{tl_example1}

**Format**

A data frame with 150 rows and 20 variables:
Index

*Topic **datasets**
  - `tl_example1`, 32
  - `.get_Dx`, 4
  - `.get_H`, 10, 31
  - `.get_L`, 12, 31
  - `.get_Pa`, 10, 14
  - `.get_Rabs`, 16, 31
  - `.get_Sr`, 19, 31
  - `.get_Tv`, 20
  - `.get_dwv`, 2
  - `.get_gbw`, 5, 8
  - `.get_gh`, 6, 10
  - `.get_gr`, 7
  - `.get_gtw`, 8
  - `.get_hvap`, 11
  - `.get_nu`, 13
  - `.get_ps`, 15
  - `.get_re`, 17
  - `.get_sh`, 18

  `Ar`, 22
  
  calculated, 3, 5–7, 10, 12, 14, 18, 19, 21, 22, 28
  
  constants, 23, 28
  
  convert_conductance, 23

  E, 24
  
  energy_balance, 25
  
  enviro_par, 26, 28

  future_map, 30

  leaf_par, 27, 28

  make_constants (make_parameters), 27
  
  make_enviropar (make_parameters), 27
  
  make_leafpar, 9
  
  make_leafpar (make_parameters), 27
  
  make_parameters, 27

parameter_names, 29

  tealeaves, 29, 32
  
  tl_example1, 32
  
  tleaf (tleaves), 30
  
  tleaves, 30

unrout, 26

33