Package ‘photosynthesis’

July 1, 2020

Version 1.0.2

Date 2020-06-30

Title Model C3 Photosynthesis

Depends R (>= 3.5.0), units (>= 0.6.0)

Imports checkmate (>= 1.9.0), crayon (>= 1.3.0), dplyr (>= 0.8.0), furrr (>= 0.1.0), future (>= 1.10.0), glue (>= 1.3.0), gunit (>= 1.0.0), magrittr (>= 1.5.0), methods (>= 3.5.0), purrr (>= 0.2.5), rlang (>= 0.3.0), stringr (>= 1.3.0), tealeaves (>= 1.0.5), tidyselect (>= 0.2.5)

Suggests ggplot2, knitr, rmarkdown, tidyr, testthat

Description Simulate C3 photosynthesis using the Farquhar, von Caemmerer, Berry (1980) <doi:10.1007/BF00386231> model as described in Buckley and Diaz-Espejo (2015) <doi:10.1111/pce.12459>. It uses units to ensure that parameters are properly specified and transformed before calculations. Temperature response functions get automatically ``baked'' into all parameters based on leaf temperature following Bernacchi et al. (2002) <doi:10.1104/pp.008250>. The package includes boundary layer, cuticular, stomatal, and mesophyll conductances to CO2, which each can vary on the upper and lower portions of the leaf. Use straightforward functions to simulate photosynthesis over environmental gradients such as Photosynthetic Photon Flux Density (PPFD) and leaf temperature, or over trait gradients such as CO2 conductance or photochemistry.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1

VignetteBuilder knitr

NeedsCompilation no

Author Chris Muir [aut, cre, cph] (<https://orcid.org/0000-0003-2555-3878>)

Maintainer Chris Muir <cdmuir@hawaii.edu>

Repository CRAN

Date/Publication 2020-07-01 09:40:02 UTC

R topics documented:

`photosynthesis-package` `photosynthesis` *package*

Description

Model C3 Photosynthesis

Details

See the README on GitHub

`A_supply` *CO2 supply and demand function (mol / m^2 s)*

Description

This function is not intended to be called by users directly.

Usage

```
A_supply(C_chl, pars, unitless = FALSE)

A_demand(C_chl, pars, unitless = FALSE)
```

Arguments

`C_chl`	Chloroplastic CO2 concentration in Pa of class `units`
`pars`	Concatenated parameters (`leaf_par`, `enviro_par`, and `constants`)
`unitless`	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.

Details

Supply function:

$$A = g_{\mathrm{tc}}(C_{\mathrm{air}} - C_{\mathrm{chl}})$$

Demand function:

$$A = (1 - \Gamma * / C_{\mathrm{chl}})\mathrm{min}(W_{\mathrm{carbox}}, W_{\mathrm{regen}}, W_{\mathrm{tpu}}) - R_{\mathrm{d}}$$

Symbol	*R*	*Description*	*Units*	*Default*
A	`A`	photosynthetic rate	μmol CO2 / (m^2 s)	calculated
g_{tc}	`g_tc`	total conductance to CO2	μmol CO2 / (m^2 s Pa)	calculated
C_{air}	`C_air`	atmospheric CO2 concentration	Pa	41
C_{chl}	`C_chl`	chloroplastic CO2 concentration	Pa	calculated
R_{d}	`R_d`	nonphotorespiratory CO2 release	μmol CO2 / (m^2 s)	2
$\Gamma *$	`gamma_star`	chloroplastic CO2 compensation point	Pa	3.743

Value

Value in mol / (m^2 s) of class `units`

Examples

```
bake_par <- make_bakepar()
constants <- make_constants(use_tealeaves = FALSE)
enviro_par <- make_enviropar(use_tealeaves = FALSE)
leaf_par <- make_leafpar(use_tealeaves = FALSE)
leaf_par <- bake(leaf_par, bake_par, constants)
# Or bake with piping (need library(magrittr))
# leaf_par %<>% bake(bake_par, constants)
enviro_par$T_air <- leaf_par$T_leaf

pars <- c(leaf_par, enviro_par, constants)
C_chl <- set_units(35, "Pa")

A_supply(C_chl, pars)

A_demand(C_chl, pars)
```

bake *Leaf parameter temperature responses*

Description

'bake' leaf parameters using temperature response functions

Constructor function for baked class. This will also inherit class leaf_par. This function ensures that temperature is "baked in" to leaf parameter calculations T_leaf using temperature response functions detailed below.

Usage

```
bake(leaf_par, bake_par, constants, assert_units = TRUE)

temp_resp1(par25, E_a, R, T_leaf, T_ref, unitless)

temp_resp2(par25, D_s, E_a, E_d, R, T_leaf, T_ref, unitless)
```

Arguments

leaf_par A list of leaf parameters inheriting class leaf_par. This can be generated using the make_leafpar function.

bake_par A list of temperature response parameters inheriting class bake_par. This can be generated using the make_bakepar function.

constants A list of physical constants inheriting class constants. This can be generated using the make_constants function.

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

par25 Parameter value at 25 °C of class units.

E_a Empirical temperature response value in J/mol of class units.

R Ideal gas constant in J / (mol K) of class units. See make_constants.

T_leaf Leaf temperature in K of class units. Will be converted to °C.

T_ref Reference temperature in K of class units.

unitless 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.

D_s Empirical temperature response value in J / (mol K) of class units.

E_d Empirical temperature response value in J/mol of class units.

Details

Several leaf parameters (`leaf_par`) are temperature sensitive. Temperature-sensitive parameters are input at a reference temperature of 25 °C. These parameters are provided as `par_name25` and then "baked" using the appropriate temperature response function and parameters in `bake_par`. The "baked" parameter will have the name without "25" appended (`par_name`). E.g. `V_cmax25` becomes `V_cmax`.

Temperature response functions following Bernacchi et al. 2002.

Temperature response function 1 (`temp_response1`):

$$\mathrm{par}(T_{\mathrm{leaf}}) = \mathrm{par25}\exp(E_{\mathrm{a}}/(RT_{\mathrm{ref}})(T_{\mathrm{leaf}} - 25)/(T_{\mathrm{leaf}} + 273.15))$$

T_{ref} is the reference temperature in K
T_{leaf} is the leaf temperature in °C

Temperature response function 2 (`temp_response2`) is the above equation multiplied by:

$$(1 + \exp((D_{\mathrm{s}}/R - E_{\mathrm{d}}/(RT_{\mathrm{ref}}))))/(1 + \exp((D_{\mathrm{s}}/R) - (E_{\mathrm{d}}/(R(T_{\mathrm{leaf}} + 273.15)))))$$

Function 1 increases exponentially with temperature; Function 2 peaks a particular temperature.

References

Bernacchi CJ, Portis AR, Nakano H, von Caemmerer S, Long SP. 2002. Temperature response of mesophyll conductance. Implications for the determination of Rubisco enzyme kinetics and for limitations to photosynthesis in vivo. Plant Physiology 130: 1992-8.

Examples

```
bake_par <- make_bakepar()
constants <- make_constants(use_tealeaves = FALSE)
leaf_par <- make_leafpar(replace = list(T_leaf = set_units(293.15, "K")),
                         use_tealeaves = FALSE)
baked_leafpar <- bake(leaf_par, bake_par, constants)

baked_leafpar$V_cmax25
baked_leafpar$V_cmax
```

`baked-class` *S3 class baked*

Description

See `bake`

`bake_par` *S3 class bake_par*

Description

Constructor function for bake_par class. This function ensures that leaf temperature gets properly "baked" into leaf parameters.

Usage

```
bake_par(.x)
```

Arguments

`.x`	A list to be constructed into **bake_par**.

`CO2_conductance` *Conductance to CO2 (umol / (m^2 s Pa))*

Description

Conductance to CO2 (umol / (m^2 s Pa))

- g_tc: total conductance to CO2

- g_uc: cuticular conductance to CO2

- g_bc: boundary layer conductance to CO2

- g_mc: mesophyll conductance to CO2

- g_sc: stomatal conductance to CO2

Usage

```
.get_gtc(pars, unitless)

.get_guc(pars, surface, unitless)

.get_gbc(pars, surface, unitless)

.get_gmc(pars, surface, unitless)

.get_gsc(pars, surface, unitless)
```

Arguments

`pars`	Concatenated parameters (`leaf_par`, `enviro_par`, and `constants`)
`unitless`	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.
`surface`	Leaf surface (lower or upper)

Details

Total conductance to CO2 is the sum of parallel conductances on the lower ($g_{\text{c,lower}}$) and upper ($g_{\text{c,upper}}$) leaf portions:

$$g_{\text{c,total}} = g_{\text{c,lower}} + g_{\text{c,upper}}$$

Each partial conductance consists of two parallel conductances, the cuticular conductance ($g_{\text{u,c}}$) and the in-series conductances through mesophyll ($g_{\text{m,c}}$), stomata ($g_{\text{s,c}}$), and boundary layer ($g_{\text{b,c}}$). To simplify the formula, I use substitute resistance where $r_x = 1/g_x$. For surface i:

$$g_{\text{c},i} = g_{\text{u},i} + (1/(r_{\text{m},i} + r_{\text{s},i} + r_{\text{b},i}))$$

The cuticular, stomatal, and mesophyll conductances can be the same or different for upper and lower. The partitioning factors (k_x) divide the conductance between surfaces while keeping the total conductance constant:

$$g_{x,\text{lower}} = g_x(1/(1 + k_x))$$

$$g_{x,\text{upper}} = g_x(k_x/(1 + k_x))$$

$$g_x = g_{x,\text{lower}} + g_{x,\text{upper}}$$

How the partitioning factors work:

k_x	description
0	all conductance on **lower** surface/portion
0.5	2/3 conductance on **lower** surface
1	conductance evenly divided between surfaces/portions
2	2/3 conductance on **upper** surface
Inf	all conductance on **upper** surface/portion

The boundary layer conductances for each are calculated on the basis of mass and heat transfer (see `.get_gbc`).

Symbol	*R*	*Description*	*Units*	*Default*
g_{mc}	`g_mc`	mesophyll conductance to CO2 (T_leaf)	μmol CO2 / (m^2 s Pa)	calculated
g_{sc}	`g_sc`	stomatal conductance to CO2	μmol CO2 / (m^2 s Pa)	4
g_{uc}	`g_uc`	cuticular conductance to CO2	μmol CO2 / (m^2 s Pa)	0.1
k_{mc}	`k_mc`	partition of g_{mc} to lower mesophyll	none	1

k_{sc}	`k_sc`	partition of g_{sc} to lower surface	none	1
k_{uc}	`k_uc`	partition of g_{uc} to lower surface	none	1

`constants` *S3 class constants*

Description

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

Usage

```
constants(.x, use_tealeaves)
```

Arguments

`.x` A list to be constructed into **constants**.

`use_tealeaves` Logical. Should leaf energy balance be used to calculate leaf temperature (T_leaf)? If TRUE, `tleaf` calculates T_leaf. If FALSE, user-defined T_leaf is used. Additional parameters and constants are required, see `make_parameters`.

`enviro_par` *S3 class enviro_par*

Description

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

Usage

```
enviro_par(.x, use_tealeaves)
```

Arguments

`.x` A list to be constructed into **enviro_par**.

`use_tealeaves` Logical. Should leaf energy balance be used to calculate leaf temperature (T_leaf)? If TRUE, `tleaf` calculates T_leaf. If FALSE, user-defined T_leaf is used. Additional parameters and constants are required, see `make_parameters`.

FvCB *Farquhar-von Caemmerer-Berry (FvCB) C3 photosynthesis model*

Description

Farquhar-von Caemmerer-Berry (FvCB) C3 photosynthesis model

Rubisco-limited assimilation rate

RuBP regeneration-limited assimilation rate

TPU-limited assimilation rate

Usage

```
FvCB(C_chl, pars, unitless = FALSE)

W_carbox(C_chl, pars, unitless = FALSE)

W_regen(C_chl, pars, unitless = FALSE)

W_tpu(C_chl, pars, unitless = FALSE)
```

Arguments

`C_chl`	Chloroplastic CO2 concentration in Pa of class `units`
`pars`	Concatenated parameters (`leaf_par`, `enviro_par`, and `constants`)
`unitless`	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.

Details

Equations following Buckley and Diaz-Espejo (2015):

Rubisco-limited assimilation rate:

$$W_{\mathrm{carbox}} = V_{\mathrm{c,max}} C_{\mathrm{chl}} / (C_{\mathrm{chl}} + K_{\mathrm{m}})$$

where:

$$K_{\mathrm{m}} = K_{\mathrm{C}}(1 + O/K_{\mathrm{O}})$$

RuBP regeneration-limited assimilation rate:

$$W_{\mathrm{regen}} = J C_{\mathrm{chl}} / (4 C_{\mathrm{chl}} + 8\Gamma*)$$

where J is a function of PPFD, obtained by solving the equation:

$$0 = \theta_J J^2 - J(J_{\max} + \phi_J PPFD) + J_{\max}\phi_J PPFD$$

TPU-limited assimilation rate:

$$W_{\text{tpu}} = 3V_{\text{tpu}}C_{\text{chl}}/(C_{\text{chl}} - \Gamma*)$$

Symbol	*R*	*Description*	*Units*	*Default*
C_{chl}	`C_chl`	chloroplastic CO2 concentration	Pa	input
$\Gamma*$	`gamma_star`	chloroplastic CO2 compensation point (T_leaf)	Pa	calculated
$J_{\max}$	`J_max`	potential electron transport (T_leaf)	μmol CO2 / (m^2 s)	calculated
K_{C}	`K_C`	Michaelis constant for carboxylation (T_leaf)	μmol / mol	calculated
K_{O}	`K_O`	Michaelis constant for oxygenation (T_leaf)	μmol / mol	calculated
O	`O`	atmospheric O2 concentration	kPa	21.27565
ϕ_J	`phi_J`	initial slope of the response of J to PPFD	none	0.331
PPFD	`PPFD`	photosynthetic photon flux density	umol quanta / (m^2 s)	1500
R_{d}	`R_d`	nonphotorespiratory CO2 release (T_leaf)	μmol CO2 / (m^2 s)	calculated
θ_J	`theta_J`	curvature factor for light-response curve	none	0.825
$V_{\text{c,max}}$	`V_cmax`	maximum rate of carboxylation (T_leaf)	μmol CO2 / (m^2 s)	calculated
V_{tpu}	`V_tpu`	rate of triose phosphate utilization (T_leaf)	μmol CO2 / (m^2 s)	calculated

Value

A list of four values with units umol CO2 / (m^2 s) of class `units`:
- `W_carbox`: Rubisco-limited assimilation rate
- `W_regen`: RuBP regeneration-limited assimilation rate
- `W_tpu`: TPU-limited assimilation rate
- `A`: minimum of W_carbox, W_regen, and W_tpu

References

Buckley TN and Diaz-Espejo A. 2015. Partitioning changes in photosynthetic rate into contributions from different variables. Plant, Cell & Environment 38: 1200-11.

Farquhar GD, Caemmerer S, Berry JA. 1980. A biochemical model of photosynthetic CO2 assimilation in leaves of C3 species. Planta 149: 78–90.

Examples

```
bake_par <- make_bakepar()
constants <- make_constants(use_tealeaves = FALSE)
enviro_par <- make_enviropar(use_tealeaves = FALSE)
leaf_par <- make_leafpar(use_tealeaves = FALSE)
leaf_par <- bake(leaf_par, bake_par, constants)
```

```
pars <- c(leaf_par, enviro_par, constants)
C_chl <- set_units(24.28, "Pa")
FvCB(C_chl, pars)
```

gw2gc *Convert g_c (μmol CO2/m^2/s/Pa) to g_w (μmol H2O /m^2/s/Pa)*

Description

Convert g_c (μmol CO2/m^2/s/Pa) to g_w (μmol H2O /m^2/s/Pa)

Convert g_c (umol CO2/m^2/s/Pa) to g_w (umol H2O /m^2/s/Pa)

Usage

```
gw2gc(g_w, D_c, D_w, unitless)

gc2gw(g_c, D_c, D_w, unitless)
```

Arguments

`g_w`	conductance to water vapor in units (μmol H2O / (m^2 s Pa)) of class `units`.
`D_c`	diffusion coefficient for CO2 in air in units of m^2/s of call `units`
`D_w`	diffusion coefficient for H2O in air in units of m^2/s of call `units`
`unitless`	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.
`g_c`	conductance to CO2 in units (μmol H2O / (m^2 s Pa)) of class `units`.

Details

Diffusive conductance to CO2 is generally about ~1.6x that of H2O because of the higher molecular weight. To convert, multiply conductance by the ratio of diffusion coefficients:

$$g_{\mathrm{c}} = g_{\mathrm{w}} D_{\mathrm{c}} / D_{\mathrm{w}}$$

$$g_{\mathrm{w}} = g_{\mathrm{c}} D_{\mathrm{w}} / D_{\mathrm{c}}$$

Value

Value with units μmol / (m^2 s Pa) of class `units`.

Note

This function will soon be moving to the standalone gunit package.

Examples

```
D_c <- set_units(1.29e-05, "m^2/s")
D_w <- set_units(2.12e-05, "m^2/s")
g_c <- set_units(3, "umol/m^2/s/Pa")
g_w <- gc2gw(g_c, D_c, D_w, unitless = FALSE)
g_w

gw2gc(g_w, D_c, D_w, unitles = FALSE)
```

J *J: Rate of electron transport (umol/m^2/s)*

Description

Calculate the rate of electron transport as a function of photosynthetic photon flux density (PPFD).

Usage

```
J(pars, unitless = FALSE)
```

Arguments

`pars`	Concatenated parameters (`leaf_par`, `enviro_par`, and `constants`)
`unitless`	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.

Details

J as a function of PPFD is the solution to the quadratic expression:

$$0 = \theta_J J^2 - J(J_{\text{max}} + \phi_J PPFD) + J_{\text{max}}\phi_J PPFD$$

Symbol	*R*	*Description*	*Units*	*Default*
J_{max}	`J_max`	potential electron transport (T_leaf)	μmol CO2 / (m^2 s)	calculated
ϕ_J	`phi_J`	initial slope of the response of J to PPFD	none	0.331
PPFD	`PPFD`	photosynthetic photon flux density	μmol quanta / (m^2 s)	1500
θ_J	`theta_J`	curvature factor for light-response curve	none	0.825

Value

Value in μmol/ (m^2 s) of class `units`

Examples

```
library(magrittr)
library(photosynthesis)

bake_par <- make_bakepar()
constants <- make_constants(use_tealeaves = FALSE)
enviro_par <- make_enviropar(use_tealeaves = FALSE)
leaf_par <- make_leafpar(use_tealeaves = FALSE)
enviro_par$T_air <- leaf_par$T_leaf
leaf_par %<>% bake(bake_par, constants)

pars <- c(leaf_par, enviro_par, constants)
J(pars, FALSE)
```

leaf_par *S3 class leaf_par*

Description

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

Usage

```
leaf_par(.x, use_tealeaves)
```

Arguments

.x	A list to be constructed into **leaf_par**.
use_tealeaves	Logical. Should leaf energy balance be used to calculate leaf temperature (T_leaf)? If TRUE, tleaf calculates T_leaf. If FALSE, user-defined T_leaf is used. Additional parameters and constants are required, see make_parameters.

make_parameters *Make lists of parameters for* photosynthesis

Description

Make lists of parameters for photosynthesis

make_leafpar

make_enviropar

make_bakepar

make_constants

Usage

```
make_leafpar(replace = NULL, use_tealeaves)

make_enviropar(replace = NULL, use_tealeaves)

make_bakepar(replace = NULL)

make_constants(replace = NULL, use_tealeaves)
```

Arguments

`replace`	A named list of parameters to replace defaults. If `NULL`, defaults will be used.
`use_tealeaves`	Logical. Should leaf energy balance be used to calculate leaf temperature (T_leaf)? If TRUE, `tleaf` calculates T_leaf. If FALSE, user-defined T_leaf is used. Additional parameters and constants are required, see `make_parameters`.

Details

Constants:

Symbol	*R*	*Description*	*Units*	*Default*
$D_{c,0}$	`D_c0`	diffusion coefficient for CO2 in air at 0 °C	m^2 / s	1.29e-5
$D_{h,0}$	`D_h0`	diffusion coefficient for heat in air at 0 °C	m^2 / s	1.90e-5
$D_{m,0}$	`D_m0`	diffusion coefficient for momentum in air at 0 °C	m^2 / s	1.33e-5
$D_{w,0}$	`D_w0`	diffusion coefficient for water vapor in air at 0 °C	m^2 / s	2.12e-5
ϵ	`epsilon`	ratio of water to air molar masses	none	0.622
G	`G`	gravitational acceleration	m / s^2	9.8
eT	`eT`	exponent for temperature dependence of diffusion	none	1.75
R	`R`	ideal gas constant	J / (mol K)	8.3144598
σ	`s`	Stephan-Boltzmann constant	W / (m^2 K^4)	5.67e-08
Sh	`Sh`	Sherwood number	none	calculated

Baking (i.e. temperature response) parameters:

Symbol	*R*	*Description*	*Units*	*Default*
$D_{\mathrm{s,gmc}}$	`Ds_gmc`	empirical temperature response parameter	J / (mol K)	487.29
$D_{\mathrm{s,Jmax}}$	`Ds_Jmax`	empirical temperature response parameter	J / (mol K)	388.04
$E_{\mathrm{a},\Gamma*}$	`Ea_gammastar`	empirical temperature response parameter	J / mol	24459.97
$E_{\mathrm{a,gmc}}$	`Ea_gmc`	empirical temperature response parameter	J / mol	68901.56
$E_{\mathrm{a,Jmax}}$	`Ea_Jmax`	empirical temperature response parameter	J / mol	56095.18
$E_{\mathrm{a,KC}}$	`Ea_KC`	empirical temperature response parameter	J / mol	80989.78
$E_{\mathrm{a,KO}}$	`Ea_KO`	empirical temperature response parameter	J / mol	23719.97
$E_{\mathrm{a,Rd}}$	`Ea_Rd`	empirical temperature response parameter	J / mol	40446.75
$E_{\mathrm{a,Vcmax}}$	`Ea_Vcmax`	empirical temperature response parameter	J / mol	52245.78
$E_{\mathrm{d,gmc}}$	`Ed_gmc`	empirical temperature response parameter	J / mol	148788.56
$E_{\mathrm{d,Jmax}}$	`Ed_Jmax`	empirical temperature response parameter	J / mol	121244.79

Environment parameters:

Symbol	*R*	*Description*	*Units*	*Default*
C_{air}	`C_air`	atmospheric CO2 concentration	Pa	41
O	`O`	atmospheric O2 concentration	kPa	21.27565
P	`P`	atmospheric pressure	kPa	101.3246
PPFD	`PPFD`	photosynthetic photon flux density	umol quanta / (m^2 s)	1500
RH	`RH`	relative humidity	none	0.50
u	`wind`	windspeed	m / s	2

Leaf parameters:

Symbol	*R*	*Description*	*Units*	*Default*
d	`leafsize`	leaf characteristic dimension	m	0.1
$\Gamma*$	`gamma_star`	chloroplastic CO2 compensation point (T_leaf)	Pa	calculated
$\Gamma*_{25}$	`gamma_star25`	chloroplastic CO2 compensation point (25 °C)	Pa	3.743
g_{mc}	`g_mc`	mesophyll conductance to CO2 (T_leaf)	μmol CO2 / (m^2 s Pa)	calculated
g_{mc}	`g_mc25`	mesophyll conductance to CO2 (25 °C)	μmol CO2 / (m^2 s Pa)	4
g_{sc}	`g_sc`	stomatal conductance to CO2	μmol CO2 / (m^2 s Pa)	4
g_{uc}	`g_uc`	cuticular conductance to CO2	μmol CO2 / (m^2 s Pa)	0.1
J_{max25}	`J_max25`	potential electron transport (25 °C)	μmol CO2 / (m^2 s)	200
J_{max}	`J_max`	potential electron transport (T_leaf)	μmol CO2 / (m^2 s)	calculated
k_{mc}	`k_mc`	partition of g_{mc} to lower mesophyll	none	1
k_{sc}	`k_sc`	partition of g_{sc} to lower surface	none	1
k_{uc}	`k_uc`	partition of g_{uc} to lower surface	none	1
K_{C25}	`K_C25`	Michaelis constant for carboxylation (25 °C)	μmol / mol	268.3
K_{C}	`K_C`	Michaelis constant for carboxylation (T_leaf)	μmol / mol	calculated
K_{O25}	`K_O25`	Michaelis constant for oxygenation (25 °C)	μmol / mol	165084.2
K_{O}	`K_O`	Michaelis constant for oxygenation (T_leaf)	μmol / mol	calculated
ϕ_J	`phi_J`	initial slope of the response of J to PPFD	none	0.331
R_{d25}	`R_d25`	nonphotorespiratory CO2 release (25 °C)	μmol CO2 / (m^2 s)	2
R_{d}	`R_d`	nonphotorespiratory CO2 release (T_leaf)	μmol CO2 / (m^2 s)	calculated
θ_J	`theta_J`	curvature factor for light-response curve	none	0.825
T_{leaf}	`T_leaf`	leaf temperature	K	298.15
$V_{\mathrm{c,max25}}$	`V_cmax25`	maximum rate of carboxylation (25 °C)	μmol CO2 / (m^2 s)	150
$V_{\mathrm{c,max}}$	`V_cmax`	maximum rate of carboxylation (T_leaf)	μmol CO2 / (m^2 s)	calculated
V_{tpu25}	`V_tpu25`	rate of triose phosphate utilization (25 °C)	μmol CO2 / (m^2 s)	200
V_{tpu}	`V_tpu`	rate of triose phosphate utilisation (T_leaf)	μmol CO2 / (m^2 s)	calculated

If `use_tealeaves = TRUE`, additional parameters are:

Constants:

Symbol	*R*	*Description*	*Units*	*Default*
c_p	`c_p`	heat capacity of air	J / (g K)	1.01
R_{air}	`R_air`	specific gas constant for dry air	J / (kg K)	287.058

Environmental parameters:

Symbol	*R*	*Description*	*Units*	*De*

E_q	`E_q`	energy per mole quanta	kJ / mol^2	22
f_{PAR}	`f_par`	fraction of incoming shortwave radiation that is photosynthetically active radiation (PAR)	none	0.5
r	`r`	reflectance for shortwave irradiance (albedo)	none	0.2
T_{air}	`T_air`	air temperature	K	29

Leaf parameters:

Symbol	*R*	*Description*	*Units*	*Default*
α_{l}	`abs_l`	absorbtivity of longwave radiation (4 - 80 μm)	none	0.97
α_{s}	`abs_s`	absorbtivity of shortwave radiation (0.3 - 4 μm)	none	0.50
g_{sw}	`g_sw`	stomatal conductance to H2O	(μmol H2O) / (m^2 s Pa)	converted from g_{sc}
g_{uw}	`g_uw`	cuticular conductance to H2O	(μmol H2O) / (m^2 s Pa)	converted from g_{uc}
$\text{logit}(sr)$	`logit_sr`	stomatal ratio (logit transformed)	none	converted from k_{sc}

Value

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

References

Buckley TN and Diaz-Espejo A. 2015. Partitioning changes in photosynthetic rate into contributions from different variables. Plant, Cell & Environment 38: 1200-11.

Examples

```
bake_par <- make_bakepar()
constants <- make_constants(use_tealeaves = FALSE)
enviro_par <- make_enviropar(use_tealeaves = FALSE)
leaf_par <- make_leafpar(use_tealeaves = FALSE)

leaf_par <- make_leafpar(
  replace = list(
    g_sc = set_units(3, "umol/m^2/s/Pa"),
    V_cmax25 = set_units(100, "umol/m^2/s")
  ), use_tealeaves = FALSE
)
```

`parameter_names` *Get vector of parameter names*

Description

Get vector of parameter names

Usage

```
parameter_names(which, use_tealeaves)
```

Arguments

`which`	A character string indicating which parameter names to retrieve: "leaf", "enviro", "bake", or "constants". Partial matching allowed.
`use_tealeaves`	Logical. Should leaf energy balance be used to calculate leaf temperature (T_leaf)? If TRUE, `tleaf` calculates T_leaf. If FALSE, user-defined T_leaf is used. Additional parameters and constants are required, see `make_parameters`.

Examples

```
parameter_names("leaf", use_tealeaves = FALSE)
```

`photosynthesis` *Simulate C3 photosynthesis*

Description

`photosynthesis`: simulate C3 photosynthesis over multiple parameter sets

`photo`: simulate C3 photosynthesis over a single parameter set

Usage

```
photosynthesis(
  leaf_par,
  enviro_par,
  bake_par,
  constants,
  use_tealeaves,
  progress = TRUE,
  quiet = FALSE,
  assert_units = TRUE,
  parallel = FALSE
)

photo(
  leaf_par,
  enviro_par,
  bake_par,
  constants,
  use_tealeaves,
  quiet = FALSE,
  assert_units = TRUE,
  check = TRUE,
  prepare_for_tleaf = use_tealeaves
)
```

Arguments

`leaf_par` A list of leaf parameters inheriting class `leaf_par`. This can be generated using the `make_leafpar` function.

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

`bake_par` A list of temperature response parameters inheriting class `bake_par`. This can be generated using the `make_bakepar` function.

`constants` A list of physical constants inheriting class `constants`. This can be generated using the `make_constants` function.

`use_tealeaves` Logical. Should leaf energy balance be used to calculate leaf temperature (T_leaf)? If TRUE, `tleaf` calculates T_leaf. If FALSE, user-defined T_leaf is used. Additional parameters and constants are required, see `make_parameters`.

`progress` Logical. Should a progress bar be displayed?

`quiet` Logical. Should messages be displayed?

`assert_units` Logical. Should parameter `units` be checked? 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`?

`check` Logical. Should arguments checks be done? This is intended to be disabled when `photo` is called from `photosynthesis` Default is TRUE.

`prepare_for_tleaf` Logical. Should arguments additional calculations for `tleaf`? This is intended to be disabled when `photo` is called from `photosynthesis`. Default is `use_tealeaves`.

Details

`photo`: This function takes simulates photosynthetic rate using the Farquhar-von Caemmerer-Berry (`FvCB`) model of C3 photosynthesis for single combined set of leaf parameters (`leaf_par`), environmental parameters (`enviro_par`), and physical constants (`constants`). Leaf parameters are provided at reference temperature (25 °C) and then "baked" to the appropriate leaf temperature using temperature response functions (see `bake`).

`photosynthesis`: This function uses `photo` to simulate photosynthesis over multiple parameter sets that are generated using `cross_df`.

Value

A data.frame with the following `units` columns

Input:

`C_air`	atmospheric CO2 concentration (Pa)
`g_mc25`	mesophyll conductance to CO2 at 25 °C (μmol CO2 / (m^2 s Pa))
`g_sc`	stomatal conductance to CO2 (μmol CO2 / (m^2 s Pa))
`g_uc`	cuticular conductance to CO2 (μmol CO2 / (m^2 s Pa))
`gamma_star25`	chloroplastic CO2 compensation point at 25 °C (Pa)
`J_max25`	potential electron transport at 25 °C (μmol CO2) / (m^2 s)
`K_C25`	Michaelis constant for carboxylation at 25 °C (μmol / mol)
`K_O25`	Michaelis constant for oxygenation at 25 °C (μmol / mol)
`k_mc`	partition of g_{mc} to lower mesophyll (unitless)
`k_sc`	partition of g_{sc} to lower surface (unitless)
`k_uc`	partition of g_{uc} to lower surface (unitless)
`leafsize`	leaf characteristic dimension (m)
`O`	atmospheric O2 concentration (kPa)
`P`	atmospheric pressure (kPa)
`phi_J`	initial slope of the response of J to PPFD (unitless)
`PPFD`	photosynthetic photon flux density (umol quanta / (m^2 s))
`R_d25`	nonphotorespiratory CO2 release at 25 °C (μmol CO2 / (m^2 s))
`RH`	relative humidity (unitless)
`theta_J`	curvature factor for light-response curve (unitless)
`T_air`	air temperature (K)
`T_leaf`	leaf temperature (K)
`V_cmax25`	maximum rate of carboxylation at 25 °C (μmol CO2 / (m^2 s))
`V_tpu25`	rate of triose phosphate utilization at 25 °C (μmol CO2 / (m^2 s))
`wind`	wind speed (m / s)

Baked Input:

`g_mc`	mesophyll conductance to CO2 at `T_leaf` (μmol CO2 / (m^2 s Pa))
`gamma_star`	chloroplastic CO2 compensation point at `T_leaf` (Pa)
`J_max`	potential electron transport at `T_leaf` (μmol CO2) / (m^2 s)
`K_C`	Michaelis constant for carboxylation at `T_leaf` (μmol / mol)
`K_O`	Michaelis constant for oxygenation at `T_leaf`(μmol / mol)
`R_d`	nonphotorespiratory CO2 release at `T_leaf` (μmol CO2 / (m^2 s))
`V_cmax`	maximum rate of carboxylation at `T_leaf` (μmol CO2 / (m^2 s))
`V_tpu`	rate of triose phosphate utilisation at `T_leaf` (μmol CO2 / (m^2 s))

Output:

`A`	photosynthetic rate at `C_chl` (μmol CO2 / (m^2 s))
`C_chl`	chloroplastic CO2 concentration where `A_supply` intersects `A_demand` (Pa)
`g_tc`	total conductance to CO2 at `T_leaf` (μmol CO2 / (m^2 s Pa))
`value`	`A_supply` - `A_demand` (μmol CO2 / (m^2 s)) at `C_chl`
`convergence`	convergence code (0 = converged)

Examples

```
# Single parameter set with 'photo'
```

```
bake_par <- make_bakepar()
constants <- make_constants(use_tealeaves = FALSE)
enviro_par <- make_enviropar(use_tealeaves = FALSE)
leaf_par <- make_leafpar(use_tealeaves = FALSE)
photo(leaf_par, enviro_par, bake_par, constants,
      use_tealeaves = FALSE)

# Multiple parameter sets with 'photosynthesis'

leaf_par <- make_leafpar(
  replace = list(
    T_leaf = set_units(c(293.14, 298.15), "K")
    ), use_tealeaves = FALSE
  )
photosynthesis(leaf_par, enviro_par, bake_par, constants,
               use_tealeaves = FALSE)
```

ppm2pa *Convert pressure from PPM to Pascals*

Description

Convert pressure from PPM to Pascals

Usage

```
ppm2pa(ppm, P)
```

Arguments

ppm	Pressure value in umol/mol of class units
P	Atmospheric pressure value in kPa of class units

Details

$$\text{Press}(kPa) = \text{Press}(ppm)P(kPa)$$
$$\text{Press}(Pa) = 1000\text{Press}(kPa)$$

Value

Value in Pa of class units

Examples

```
ppm <- set_units(400, "umol/mol")
P <- set_units(101.325, "kPa")
ppm2pa(ppm, P)
```

Index