Package ‘photosynthesis’

November 8, 2022

Version 2.1.0
Date 2022-11-08
Title Tools for Plant Ecophysiology & Modeling
Depends R (>= 4.0.0), ggplot2 (>= 3.3.0), minpack.lm (>= 1.2-1), units (>= 0.6.6)
Imports checkmate (>= 2.0.0), crayon (>= 1.3.4), dplyr (>= 0.8.5), furrr (>= 0.1.0), glue (>= 1.4.0), graphics (>= 4.0.0), grDevices (>= 4.0.0), gunit (>= 1.0.2), magrittr (>= 1.5.0), methods (>= 3.5.0), nlme (>= 3.1-147), purrr (>= 0.3.3), rlang (>= 0.4.6), stats (>= 4.0.0), stringr (>= 1.4.0), tealeaves (>= 1.0.5), tibble (>= 3.0.0), utils (>= 4.0.0)
Suggests future, testthat, knitr, rmarkdown, tidyr, tidyrr, tidyselect

Description Contains modeling and analytical tools for plant ecophysiology.

MODELING: Simulate C3 photosynthesis using the Farquhar, von Caemmerer, Berry (1980) 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.

R topics documented:

- photosynthesis-package
- analyze_sensitivity
- aq_response
- A_supply
- bake
- baked-class
- bake_par
- calculated-parameters
- calculate_jmax
- CO2_conductance
- compile_data
- compute_sensitivity
- constants
- enviro_par
- fit_aci_response

License MIT + file LICENSE
Encoding UTF-8
LazyData true
RoxygenNote 7.2.1
VignetteBuilder knitr
URL https://github.com/cdmuir/photosynthesis
BugReports https://github.com/cdmuir/photosynthesis/issues
NeedsCompilation no
Author Joseph Stinziano [aut] (<https://orcid.org/0000-0002-7628-4201>), Cassandra Roback [aut], Demi Sargent [aut], Bridget Murphy [aut], Patrick Hudson [aut, dtc], Chris Muir [aut, cre] (<https://orcid.org/0000-0003-2555-3878>)
Maintainer Chris Muir <cdmuir@hawaii.edu>
Repository CRAN
Date/Publication 2022-11-08 20:20:02 UTC
**Description**

Modeling and analytical tools for plant ecophysiology

**Details**

See the README on GitHub

**analyze_sensitivity**  
Running 2-parameter sensitivity analyses
Usage

analyze_sensitivity(
    data,
    funct,
    test1 = NA,
    values1,
    test2 = NA,
    values2,
    element_out = 1,
    ...
)

Arguments

data     Dataframe
funct    Function to use - do not use parentheses
test1    Input parameter to vary and test
values1  Values of test1 to use
test2    Input parameter to vary and test
values2  Values of test2 to use
element_out List element to compile
...  Additional arguments required for the function

Value

analyze_sensitivity runs a 2-parameter sensitivity analysis. Note that any parameter value combinations that break the input function WILL break this function. For 1-parameter sensitivity analysis, use test1 only.

Examples

# Read in your data
# Note that this data is coming from data supplied by the package
# hence the complicated argument in read.csv()
# This dataset is a CO2 by light response curve for a single sunflower
data <- read.csv(system.file("extdata", "A_Ci_Q_data_1.csv", package = "photosynthesis")

# Define a grouping factor based on light intensity to split the ACi curves
data$Q_2 <- as.factor(round(data$Qin, digits = 0))

# Convert leaf temperature to K
data$T_leaf <- data$Tleaf + 273.15

# Run a sensitivity analysis on gamma_star and mesophyll conductance
# at 25 Celsius for one individual curve
# pars <- analyze_sensitivity(
#   data = data[data$Q_2 == 1500, ],
#   funct = fit_aci_response,
#   varnames = list(
#     A_net = "A",
#     T_leaf = "T_leaf",
#     C_i = "Ci",
#     PPFD = "Qin"
#   ),
#   useg_mct = TRUE,
#   test1 = "gamma_star25",
#   element_out = 1,
#   test2 = "g_mc25",
#   fitTPU = TRUE,
#   Ea_gamma_star = 0,
#   Ea_g_mc = 0,
#   values1 = seq(
#     from = 20,
#     to = 40,
#     by = 2
#   ),
#   values2 = seq(
#     from = 0.5,
#     to = 2,
#     by = 0.1
#   )
# )
# )
# Graph V_cmax
# ggplot(pars, aes(x = gamma_star25, y = g_mc25, z = V_cmax)) +
# geom_tile(aes(fill = V_cmax)) +
# labs(
#   x = expression(Gamma * "^25" ~ "(" * mu * mol ~ mol
#   ^ -1
#   ) * ")"),
#   y = expression(g[m][25] ~ "(" * mu * mol ~ m
#   ^ -2
#   ) ~ s*(
#     -1
#   ) ~ Pa*
#   ^ -1
#   ) * ")")
# ) +
# scale_fill_distiller(palette = "Greys") +
# geom_contour(colour = "Black", size = 1) +
# theme_bw()
**aq_response**

*Non-rectangular hyperbolic model of light responses*

**Description**

Non-rectangular hyperbolic model of light responses

**Usage**

\[ aq\_response(k\_sat, \phi\_J, Q\_abs, \theta\_J) \]

**Arguments**

- **k_sat**: Light saturated rate of process \( k \)
- **phi_J**: Quantum efficiency of process \( k \)
- **Q_abs**: Absorbed light intensity (\( \text{umol m}^{-2} \text{s}^{-1} \))
- **theta_J**: Curvature of the light response

**Value**

\( aq\_response \) is used to describe the response of a process to absorbed light intensity. Assumes that input is absorbed light. Note that if absorbed light is not used, then the meaning of \( \phi\_J \) becomes unclear. This function is designed to be used with \( \text{fit\_aq\_response} \), however it could easily be fed into a different fitting approach (e.g. Bayesian approaches). Originally from Marshall et al. 1980.

**References**


---

**A_supply**

*CO2 supply and demand function (\( \text{mol} / \text{m}^2 \text{s} \))*

**Description**

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

**Usage**

\[ A\_supply(C\_chl, \text{pars}, \text{unitless} = \text{FALSE}, \text{use\_legacy\_version} = \text{FALSE}) \]

\[ A\_demand(C\_chl, \text{pars}, \text{unitless} = \text{FALSE}) \]
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.
use_legacy_version  Logical. Should legacy model (<2.1.0) be used? See NEWS for further information. Default is FALSE.

Details

Supply function:

\[ A = g_{tc}(C_{air} - C_{chl}) \]

Demand function:

\[ A = \left(1 - \Gamma^* / C_{chl}\right) \min(W_{carbox}, W_{regen}, W_{tpu}) - R_d \]

Symbol | R | Description | Units | Default
--- | --- | --- | --- | ---
A | \( A \) | photosynthetic rate | \( \mu \text{mol CO}_2 / (m^2 \text{s}) \) | calculated
\( g_{tc} \) | g\_tc | total conductance to CO2 | \( \mu \text{mol CO}_2 / (m^2 \text{s Pa}) \) | 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 | \( \mu \text{mol CO}_2 / (m^2 \text{s}) \) | 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, enviro_par, bake_par, constants)
# Or bake with piping (need library(magrittr))
# leaf_par %<>% bake(enviro_par, bake_par, constants)
enviro_par$T\_air = leaf_par$T\_leaf

pars = c(leaf_par, enviro_par, constants)
C_chl = set_units(350, umol/mol)
A_supply(C_chl, pars)
A_demand(C_chl, pars)
**bake**  
*Leaf parameter temperature responses*

**Description**

'bake' leaf parameters using temperature response functions

**Usage**

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

temp_res1(par25, E_a, R, T_leaf, T_ref, unitless)

temp_res2(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.
- `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.
- `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 Buckley and Diaz-Espejo (2015)

Temperature response function 1 (temp_response1):

\[
par(T_{leaf}) = par25 \exp\left(\frac{E_a}{RT_{ref}}(T_{leaf} - 25)/(T_{leaf} + 273.15)\right)
\]

\(T_{ref}\) 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:

\[
\frac{(1 + \exp((D_s/R - E_d/(RT_{ref}))))/(1 + \exp((D_s/R) - (E_d/(R(T_{leaf} + 273.15)))))}{(1 + \exp((D_s/R - E_d/(RT_{ref}))))/(1 + \exp((D_s/R) - (E_d/(R(T_{leaf} + 273.15))))))}
\]

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

Value

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

References


Examples

```r
bake_par = make_bakepar()
constants = make_constants(use_tealeaves = FALSE)
enviro_par = make_enviropar(use_tealeaves = FALSE)
leaf_par = make_leafpar(
    replace = list(T_leaf = set_units(293.15, K)),
    use_tealeaves = FALSE
)
baked_leafpar = bake(leaf_par, enviro_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

S3 class `bake_par`

#### Usage

`bake_par(.x)`

#### Arguments

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

#### Value

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

### calculated-parameters

Get default functions for calculated parameters in *photosynthesis*

#### Description

Get default functions for calculated parameters in *photosynthesis*

#### Usage

`get_f_parameter(.f_name)`

#### Arguments

- `.f_name` character string of function
calculate_jmax

Inverse non-rectangular hyperbola for J_max calculation

Description

Inverse non-rectangular hyperbola for J_max calculation

Usage

 calculate_jmax(PPFD, alpha, J, theta_J)

 calculate_j(PPFD, alpha, J_max, theta_J)

Arguments

 PPFD light intensity in umol m^-2 s^-1
 alpha initial slope of the light response
 J electron transport rate in umol m^-2 s^-1
 theta_J curvature of the light response
 J_max maximum rate of electron transport in umol m^-2 s^-1

Value

calculate_jmax calculates J_max given PPFD and J. It is necessary for the electron transport component of the fit_aci_response function.

calculate_j provides a model of the light response of J. It is necessary for fitting the electron transport component of the photosynthetic CO2 response curves in fit_aci_response.

CO2_conductance

Conductance to CO2 (mol / m^2 / s)

Description

Conductance to CO2 (mol / m^2 / s)

 - 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, use_legacy_version)`
- `get_guc(pars, surface, unitless)`
- `get_gbc(pars, surface, unitless, use_legacy_version)`
- `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.
- **use_legacy_version**
  Logical. Should legacy model (<2.1.0) be used? See NEWS for further information. Default is FALSE.
- **surface**
  Leaf surface (lower or upper)

Details

**Default conductance model**

The conductance model described in this section is used by default unless additional anatomical parameters described in the next section are provided.

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

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

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

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

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:

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

How the partitioning factors work:
The boundary layer conductances for each are calculated on the basis of mass and heat transfer (see .get_gbc).

<table>
<thead>
<tr>
<th>Symbol</th>
<th>R</th>
<th>Description</th>
<th>Units</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>( g_{mc} )</td>
<td>g_{mc}</td>
<td>mesophyll conductance to CO2 (T_leaf)</td>
<td>mol / m(^2) / s</td>
<td>calculated</td>
</tr>
<tr>
<td>( g_{sc} )</td>
<td>g_{sc}</td>
<td>stomatal conductance to CO2</td>
<td>mol / m(^2) / s</td>
<td>0.4</td>
</tr>
<tr>
<td>( g_{uc} )</td>
<td>g_{uc}</td>
<td>cuticular conductance to CO2</td>
<td>mol / m(^2) / s</td>
<td>0.01</td>
</tr>
<tr>
<td>( k_{mc} )</td>
<td>k_{mc}</td>
<td>partition of ( g_{uc} ) to lower mesophyll</td>
<td>none</td>
<td>1</td>
</tr>
<tr>
<td>( k_{sc} )</td>
<td>k_{sc}</td>
<td>partition of ( g_{sc} ) to lower surface</td>
<td>none</td>
<td>1</td>
</tr>
<tr>
<td>( k_{uc} )</td>
<td>k_{uc}</td>
<td>partition of ( g_{uc} ) to lower surface</td>
<td>none</td>
<td>1</td>
</tr>
</tbody>
</table>

New conductance model

The conductance model described in this section is implemented in photosynthesis (>= 2.1.0) if parameters to calculate the internal airspace and liquid-phase conductances (\( A_{mes,A} \), \( g_{liq} \)) are provided. These parameters are 1) the effective path lengths through the lower and upper leaf internal airspaces (\( \delta_{ias,lower} \), \( \delta_{ias,upper} \)) and 2) the mesophyll area per leaf area (\( A_{mes,A} \)) and liquid-phase conductance per mesophyll cell area (\( g_{liq} \)).

Two parallel diffusion pathways, one from each leaf surface, converge to a single CO2 concentration at the mesophyll cell boundary. We use a single liquid-phase resistance to represent the combined cell wall, plasmalemma, and chloroplast resistances. The gas-phase resistance through boundary layer, cuticle/stomata, and internal airspace is \( r_{gas,c} \); the liquid-phase intracellular resistance is \( r_{i,c} \).

\[
\frac{1}{r_{total,c}} = \frac{1}{r_{gas,c}} + \frac{1}{r_{i,c}}
\]

The gas-phase resistance occurs through two parallel pathways, which we refer to as the 'lower' and 'upper' pathways because horizontally oriented leaves often have different anatomical properties on each surface. The gas-phase resistance through pathway \( i \in \{\text{lower, upper}\} \) is:

\[
r_{gas,c,i} = r_{b,c,i} + r_{u+s,c,i} + r_{ias,c,i}
\]

The subscripts \( b, u+s, \) and \( ias \) denote boundary layer, cuticular + stomatal, and internal airspace, respectively. The subscript \( c \) indicates we are considering the conductance to CO2 rather than another molecular species.

Cuticular and stomatal conductances (1 / resistance) are parallel, so:

\[
\frac{1}{r_{u+s,c,i}} = g_{u+s,c,i} = g_{u,c,i} + g_{s,c,i}
\]

Substituting the above expression into the equation for \( r_{gas,c,i} \):
The total gas-phase resistance is the inverse of the sum of the parallel lower and upper conductances:

\[ \frac{1}{r_{\text{gas,c}}} = g_{\text{gas,c,lower}} + g_{\text{gas,c,upper}} \]

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

\[ g_x,\text{lower} = g_x \left( \frac{1}{1 + k_x} \right) \]
\[ g_x,\text{upper} = g_x \left( \frac{k_x}{1 + k_x} \right) \]
\[ g_x = g_x,\text{lower} + g_x,\text{upper} \]

How the partitioning factors work:

\[ k_x \quad \text{description} \]
\[ 0 \quad \text{all conductance on lower surface/portion} \]
\[ 0.5 \quad \text{2/3 conductance on lower surface} \]
\[ 1 \quad \text{conductance evenly divided between surfaces/portions} \]
\[ 2 \quad \text{2/3 conductance on upper surface} \]
\[ \text{Inf} \quad \text{all conductance on upper surface/portion} \]

The internal airspace conductance is the diffusivity of CO2 at a given temperature and pressure divided by the effective path length:

\[ g_{\text{iasc,lower}} = \frac{D_c}{\delta_{\text{iasc,lower}}} \]
\[ g_{\text{iasc,upper}} = \frac{D_c}{\delta_{\text{iasc,upper}}} \]

\( g_{\text{iasc,lower}} \) and \( g_{\text{iasc,upper}} \) are calculated in the \texttt{bake} function. See \texttt{.get_Dx} for calculating \( D_c \).

The liquid-phase intracellular resistance is given by:

\[ \frac{1}{r_{i,c}} = g_{i,c} = \frac{g_{\text{liq,c}}A_{\text{mes}}}{A} \]

\( g_{\text{liq,c}} \) is temperature sensitive. See \texttt{bake}.

The boundary layer conductances for each are calculated on the basis of mass and heat transfer (see \texttt{.get_gbc}).
**Description**

Compiling outputs from lists

**Usage**

`compile_data(data, output_type = "list", list_element)`

**Arguments**

- `data` List of elements
- `output_type` Type of desired output. For graphs or models, use "list", for parameters, use "dataframe".
- `list_element` Which elements of the sublists do you wish to compile?

**Value**

`compile_data` converts the outputs of `fit_many` into a form more readily usable for analysis. Can be used to create dataframe of all fitted parameters, a list of model outputs, a list of graphs for plotting. This function is NOT restricted to compiling outputs from plantecophystools but could be used to compile elements from ANY list of lists.

**Examples**

```r
# Read in your data
# Note that this data is coming from data supplied by the package
# hence the complicated argument in read.csv()
# This dataset is a CO2 by light response curve for a single sunflower
data <- read.csv(system.file("extdata", "A_Ci_Q_data_1.csv", package = "photosynthesis")

# Define a grouping factor based on light intensity to split the ACi curves
data$Q_2 <- as.factor(round(data$Qin, digits = 0))

# Convert leaf temperature to K
data$T_leaf <- data$T_leaf + 273.15

# Fit many curves
fits <- fit_many(
  data = data,
  varnames = list(
    A_net = "A",
```
function

funct = fit_aci_response,


group = "Q_2"

# Compile graphs into a list for plotting
fits_graphs <- compile_data(fits,

list_element = 2

)

# Plot one graph from the compiled list
plot(fits_graphs[[1]])

compute_sensitivity

### compute_sensitivity

#### Computing measures of sensitivity

**Description**

Computing measures of sensitivity

**Usage**

```r
compute_sensitivity(
  data,
  varnames = list(Par = "Par", test1 = "test1", test2 = "test2"),
  test1_ref,
  test2_ref
)
```

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>data</td>
<td>Dataframe with output from sensitivity_analysis()</td>
</tr>
<tr>
<td>varnames</td>
<td>Variable names</td>
</tr>
<tr>
<td>test1_ref</td>
<td>Reference value for parameter</td>
</tr>
<tr>
<td>test2_ref</td>
<td>Reference value for parameter</td>
</tr>
</tbody>
</table>

**Value**

compute_sensitivity calculates two sets of sensitivity measures: parameter effect (Bauerle et al., 2014), and control coefficient (Capaldo & Pandis, 1997). This function is useful in determining how much a given input (assumed or otherwise) can affect the model output and conclusions. Particularly useful if a given parameter is unknown during a fitting or modeling process.
compute_sensitivity

References


Examples

# Read in your data
# Note that this data is coming from data supplied by the package
# hence the complicated argument in read.csv()
# This dataset is a CO2 by light response curve for a single sunflower
data <- read.csv(system.file("extdata", "A_Ci_Q_data_1.csv",  
                  package = "photosynthesis")
))

# Define a grouping factor based on light intensity to split the ACi curves
data$Q_2 <- as.factor(round(data$Qin, digits = 0)))

# Convert leaf temperature to K
data$T_leaf <- data$Tleaf + 273.15

# Run a sensitivity analysis on gamma_star and mesophyll conductance
# at 25 Celsius for one individual curve
# pars <- analyze_sensitivity(  
#   data = data[data$Q_2 == 1500, ],  
#   funct = fit_aci_response,  
#   varnames = list(  
#     A_net = "A",  
#     T_leaf = "T_leaf",  
#     C_i = "Ci",  
#     PPFD = "Qin"  
#   ),  
#   useg_mct = TRUE,  
#   test1 = "gamma_star25",  
#   element_out = 1,  
#   test2 = "g_mc25",  
#   fitTPU = TRUE,  
#   Ea_gamma_star = 0,  
#   Ea_g_mc = 0,  
#   values1 = seq(  
#     from = 20,  
#     to = 60,  
#     by = 2  
#   ),  
#   values2 = seq(  
#     from = 0.2,  
#     to = 2,  
#   ),
# by = 0.1
#
# Compute measures of sensitivity
# par2 <- compute_sensitivity(
#   data = pars,
#   varnames = list(
#     Par = "V_cmax",
#     test1 = "gamma_star25",
#     test2 = "g_mc25"
#   ),
#   test1_ref = 42,
#   test2_ref = 1
# )
#)
# Plot control coefficients
# ggplot(par2, aes(y = CE_gamma_star25, x = CE_g_mc25, colour = V_cmax)) +
# geom_point() +
# theme_bw()
# # Note that in this case a missing point appears due to an infinity

---

**constants**

**S3 class constants**

**Description**

S3 class constants

**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`)?
  
**Value**

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

S3 class enviro_par

Description

S3 class enviro_par

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.

Value

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

fit_aci_response

Fitting ACi curves

Description

Fitting ACi curves

Usage

fit_aci_response(
  data,
  varnames = list(A_net = "A_net", T_leaf = "T_leaf", C_i = "C_i", PPFD = "PPFD", g_mc = "g_mc"),
  P = 100,
  fitTPU = TRUE,
  alpha_g = 0,
  R_d_meas = NULL,
  useR_d = FALSE,
  useg_mc = FALSE,
  useg_mct = FALSE,
  usegamma_star = FALSE,
  useK_M = FALSE,
useK_C_K_O = FALSE,
alpha = 0.24,
theta_J = 0.85,
gamma_star25 = 42.75,
Ea_gamma_star = 37830,
K_M25 = 718.4,
Ea_K_M = 65508.28,
g_mc25 = 0.08701,
Ea_g_mc = 0,
K_C25 = NULL,
Ea_K_C = NULL,
K_O25 = NULL,
Ea_K_O = NULL,
Oconc = 21,
gamma_star_set = NULL,
K_M_set = NULL,

Arguments

data Dataframe for A-Ci curve fitting

varnames List of variable names. varnames = list(A_net = "A_net", T_leaf = "T_leaf", C_i = "C_i", PPFD = "PPFD", g_mc = "g_mc"), where A_net is net CO2 assimilation, T_leaf is leaf temperature in Celsius, C_i is intercellular CO2 concentration in umol/mol, PPFD is incident irradiance in umol m-2 s-1 (note that it is ASSUMED to be absorbed irradiance, so be sure to adjust according to light absorbance and PSI/PSII partitioning accordingly OR interpret the resultant values of J and J_max with caution), g_mc is mesophyll conductance to CO2 in mol m-2 s-1 Pa-1.

P Atmospheric pressure in kPa

fitTPU Should triose phosphate utilization (V_TPU) be fit?

alpha_g Fraction of respiratory glycolate carbon that is not returned to the chloroplast (von Caemmerer, 2000). If ACi curves show high-CO2 decline, then this value should be > 0.

R_d_meas Measured value of respiratory CO2 efflux in umol m-2 s-1. Input value should be positive to work as expected with the equations.

useR_d Use a measured value of R_d? Set to TRUE if using R_d_meas.

useg_mc Use mesophyll conductance? Set to TRUE if specifying g_mc in varnames above.

useg_mct Use mesophyll conductance temperature response? Set to TRUE if using a temperature response of mesophyll conductance.

usegamma_star Specify gamma_star value? If FALSE, uses a temperature response function with Nicotiana tabacum defaults from Bernacchi et al. 2001.

fit_aci_response

useK_C_K_O Use individual carboxylation/oxygenation constants for rubisco? If TRUE, need to specify values at 25C and activation energy for the Arrhenius temperature response function.
alpha Quantum yield of CO2 assimilation
theta_J Curvature of the photosynthetic light response curve
gamma_star25 gamma_star at 25C in umol mol-1
Ea_gamma_star Activation energy of gamma_star in J mol-1
K_M25 Michaelis-Menten constant for rubisco at 25C
Ea_K_M Activation energy for K_M in J mol-1
g_mc25 Mesophyll conductance at 25C in mol m-2 s-1
Ea_g_mc Activation energy of g_mc in J mol-1
K_C25 Michaelis-Menten constant for rubisco carboxylation at 25C
Ea_K_C Activation energy for K_C in J mol-1
K_O25 Michaelis-Menten constant for rubisco oxygenation at 25C
Ea_K_O Activation energy for K_O in J mol-2
Oconc O2 concentration in intracellular O2 when using K_C_K_O
gamma_star_set Value of gamma_star to use (in ppm) if usegamma_star = TRUE
K_M_set Value of K_M to use if useK_M = TRUE
... Other arguments to pass on

Value

fit_aci_response fits ACI curves using an approach similar to Gu et al. 2010. Iterates all possible C_i transition points and checks for inadmissible curve fits. If no curves are admissible (either due to poor data or poor assumed parameters), the output will include a dataframe of NA values. Default parameters are all from Bernacchi et al. 2001, 2002.

References

Examples

# Read in your data
# Note that this data is coming from data supplied by the package
# hence the complicated argument in read.csv()
# This dataset is a CO2 by light response curve for a single sunflower
data <- read.csv(system.file("extdata", "A_Ci_Q_data_1.csv", package = "photosynthesis")
)

# Define a grouping factor based on light intensity to split the ACi curves
data$Q_2 <- as.factor(round(data$Qin, digits = 0)))

# Convert leaf temperature to K
data$T_leaf <- data$Tleaf + 273.15

# Fit ACi curve. Note that we are subsetting the dataframe here to fit for a single value of Q_2
fit <- fit_aci_response(data[data$Q_2 == 1500, ],
    varnames = list(
        A_net = "A",
        T_leaf = "T_leaf",
        C_i = "Ci",
        PPFD = "Qin"
    )
)

# View fitted parameters
fit[[1]]

# View graph
fit[[2]]

# View data with modelled parameters attached
fit[[3]]

# Fit many curves
fits <- fit_many(
data = data,
    varnames = list(
        A_net = "A",
        T_leaf = "T_leaf",
        C_i = "Ci",
        PPFD = "Qin"
    ),
    funct = fit_aci_response,
    group = "Q_2"
)

# Print the parameters
# First set of double parentheses selects an individual group value
fit_aq_response

# Second set selects an element of the sublist
fits[[3]][[1]]

# Print the graph
fits[[3]][[2]]

# Compile graphs into a list for plotting
fits_graphs <- compile_data(fits,
                      list_element = 2
)

# Compile parameters into dataframe for analysis
fits_pars <- compile_data(fits,
                      output_type = "dataframe",
                      list_element = 1
)

fit_aq_response

Fitting light responses of net CO2 assimilation

Description

Fitting light responses of net CO2 assimilation

Usage

fit_aq_response(
  data,
  varnames = list(A_net = "A_net", PPFD = "PPFD"),
  usealpha_Q = FALSE,
  alpha_Q = 0.84,
  title = NULL
)

Arguments

data Dataframe containing CO2 assimilation light response

varnames Variable names where varnames = list(A_net = "A_net", PPFD = "PPFD").
  A_net is net CO2 assimilation in umol m-2 s-1, PPFD is incident irradiance.
  PPFD can be corrected for light absorbance by using usealpha_Q and setting
  alpha_Q.

usealpha_Q Correct light intensity for absorbance? Default is FALSE.

alpha_Q Absorbance of incident light. Default value is 0.84.

title Title for graph
Value

`fit_aq_response` fits the light response of net CO2 assimilation. Output is a dataframe containing light saturated net CO2 assimilation, quantum yield of CO2 assimilation (\(\phi_J\)), curvature of the light response (\(\theta_J\)), respiration (\(R_d\)), light compensation point (LCP), and residual sum of squares (\(\text{resid}_{SS}\)). Note that \(R_d\) fitted in this way is essentially the same as the Kok method, and represents a respiration value in the light that may not be accurate. \(R_d\) output should thus be interpreted more as a residual parameter to ensure an accurate fit of the light response parameters. Model originally from Marshall et al. 1980.

References


Examples

```r
# Read in your data
# Note that this data is coming from data supplied by the package
# hence the complicated argument in read.csv()
# This dataset is a CO2 by light response curve for a single sunflower
data <- read.csv(system.file("extdata", "A_Ci_Q_data_1.csv",
  package = "photosynthesis"
))

# Fit many AQ curves
# Set your grouping variable
# Here we are grouping by CO2_s and individual
data$C_s <- (round(data$CO2_s, digits = 0))

data$C_s <- as.factor(round(data$C_s, digits = -1))

# To fit one AQ curve
fit <- fit_aq_response(data[data$C_s == 600, ],
  varnames = list(
    A_net = "A",
    PPFD = "Qin"
  )
)

# Print model summary
summary(fit[[1]])

# Print fitted parameters
fit[[2]]

# Print graph
fit[[3]]

# Fit many curves
```
fit_gs_model <- fit_many(
  data = data,
  varnames = list(
    A_net = "A",
    PPFD = "Qin",
    group = "C_s"
  ),
  funct = fit_aq_response,
  group = "C_s"
)

# Look at model summary for a given fit
# First set of double parentheses selects an individual group value
# Second set selects an element of the sublist
summary(fits[[3]][[1]])

# Print the parameters
fits[[3]][[2]]

# Print the graph
fits[[3]][[3]]

# Compile graphs into a list for plotting
fits_graphs <- compile_data(fits,
  list_element = 3
)

# Compile parameters into dataframe for analysis
fits_pars <- compile_data(fits,
  output_type = "dataframe",
  list_element = 2
)

---

fit_gs_model  Fitting stomatal conductance models

Description
Fitting stomatal conductance models

Usage
fit_gs_model(
  data,
  varnames = list(A_net = "A_net", C_air = "C_air", g_sw = "g_sw", RH = "RH", VPD = "VPD"),
  model = c("BallBerry", "Leuning", "Medlyn_partial", "Medlyn_full"),
  D0 = 3,
  ...
)
Arguments

data  
Dataframe

varnames  
Variable names

For the Ball-Berry model: varnames = list(A_net = "A_net", C_air = "C_air", 
g_sw = "g_sw", RH = "RH") where A_net is net CO2 assimilation, C_air is CO2 
concentration at the leaf surface in umol mol-1, g_sw is stomatal conductance 
to H2O, and RH is relative humidity as a proportion.

For the Leuning model: varnames = list(A_net = "A_net", C_air = "C_air", g_sw 
= "g_sw", VPD = "VPD") where A_net is net CO2 assimilation, C_air is CO2 
concentration at the leaf surface in umol mol-1, g_sw is stomatal conductance 
to H2O, and VPD is leaf to air vapor pressure deficit in kPa.

For the Medlyn et al. 2011 models: varnames = list(A_net = "A_net", C_air = 
"C_air", g_sw = "g_sw", VPD = "VPD") where A_net is net CO2 assimilation, 
C_air is CO2 concentration at the leaf surface in umol mol-1, g_sw is stomatal 
conductance to H2O, and VPD is leaf to air vapor pressure deficit in kPa.

model  
Which model(s) to fit? Defaults to all models. Available options are "Ball-
Berry", "Leuning", "Medlyn_partial", and "Medlyn_full", from Ball et al. (1987), 
Leuning (1995), and Medlyn et al. (2011).

D0  
Vapor pressure sensitivity of stomata (Leuning 1995)

...  
Arguments to pass on to the nlsLM() function for the Medlyn models.

Value

fit_gs_model fits one or more stomatal conductance models to the data. The top level of the output 
list is named after the fitted model, while the second level contains the Model, Parameters, and 
Graph, in that order.

References

Ball JT, Woodrow IE, Berry JA. 1987. A model predicting stomatal conductance and its con-	ribution to the control of photosynthesis under different environmental conditions, in Progress in 
Photosynthesis Research, Proceedings of the VII International Congress on Photosynthesis, vol. 4, 

Plant Cell Environ 18:339-357

Medlyn BE, Duursma RA, Eamus D, Ellsworth DS, Prentice IC, Barton CVM, Crous KY, Angelis 
PD, Freeman M, Wingate L. 2011. Reconciling the optimal and empirical approaches to modeling 
stomatal conductance. Glob Chang Biol 17:2134-2144

Examples

# Read in your data
# Note that this data is coming from data supplied by the package 
# hence the complicated argument in read.csv()
# This dataset is a CO2 by light response curve for a single sunflower

data <- read.csv(system.file("extdata", "A_Ci_Q_data_1.csv"),
package = "photosynthesis"
})

# Convert RH to a proportion
data$RH <- data$RHcham / 100

# Fit stomatal conductance models
# Can specify a single model, or all as below
fits <- fit_gs_model(
data = data,
varnames = list(
  A_net = "A",
  C_air = "Ca",
  g_sw = "gsw",
  RH = "RH",
  VPD = "VPDleaf"
),
model = c(
  "BallBerry",
  "Leuning",
  "Medlyn_partial",
  "Medlyn_full"
),
D0 = 3
)

# Look at BallBerry model summary:
summary(fits["BallBerry"][["Model"]])

# Look at BallBerry parameters
fits["BallBerry"][["Parameters"]]

# Look at BallBerry plot
fits["BallBerry"][["Graph"]]

# Fit many g_sw models
# Set your grouping variable
# Here we are grouping by Qin and individual
data$Q_2 <- as.factor(round(data$Qin, digits = 0))

fits <- fit_many(data,
  varnames = list(
    A_net = "A",
    C_air = "Ca",
    g_sw = "gsw",
    RH = "RH",
    VPD = "VPDleaf"
  ),
  funct = fit_gs_model,
  group = "Q_2"
)

# Look at the Medlyn_partial outputs at 750 PAR
# Model summary
summary(fits["750"]["Medlyn_partial"]["Model"])

# Model parameters
fits["750"]["Medlyn_partial"]["Parameters"]

# Graph
fits["750"]["Medlyn_partial"]["Graph"]

# Compile parameter outputs for BallBerry model
# Note that it's the first element for each PAR value
# First compile list of BallBerry fits
bbmods <- compile_data(
data = fits,
  output_type = "list",
  list_element = 1
)
# Now compile the parameters (2nd element) into a dataframe
bbpars <- compile_data(
data = bbmods,
  output_type = "dataframe",
  list_element = 2
)

# Convert group variable back to numeric
bbpars$ID <- as.numeric(bbpars$ID)

# Take quick look at light response of intercept parameters
plot(g0 ~ ID, bbpars)

# Compile graphs
graphs <- compile_data(
data = bbmods,
  output_type = "list",
  list_element = 3
)
# Look at 3rd graph
graphs[[3]]

---

**fit_g_mc_variableJ**  
*Fitting mesophyll conductance with the variable J method*

---

**Description**

Fitting mesophyll conductance with the variable J method
Usage

```r
fit_g_mc_variableJ(
data,
  varnames = list(A_net = "A_net", J_etr = "J_etr", C_i = "C_i", PPFD = "PPFD", phi_PSII = "phi_PSII"),
  usealpha_Q = FALSE,
  alpha_Q = 0.84,
  beta_Q = 0.5,
  gamma_star,
  R_d,
  P = 100
)
```

Arguments

- **data**: Dataframe
- **varnames**: Variable names to fit g_mc. varnames = list(A_net = "A_net", J_etr = "J_etr", C_i = "C_i", PPFD = "PPFD", phi_PSII = "phi_PSII"), where A_net is net CO2 assimilation in umol m-2 s-1, J_etr is linear electron transport flux in umol m-2 s-1, C_i is intercellular CO2 concentration in umol mol-1, PPFD is incident irradiance in umol m-2 s-1, phi_PSII is the operating efficiency of photosystem II.
- **usealpha_Q**: Recalculate electron transport with new absorbance value?
- **alpha_Q**: Absorbance of photosynthetically active radiation
- **beta_Q**: Partitioning of absorbed light energy between PSI and PSII
- **gamma_star**: Photorespiratory CO2 compensation point in umol mol-1
- **R_d**: Respiration rate in umol m-2 s-1
- **P**: Atmospheric pressure in kPa

Value

`fit_g_mc_variableJ` fits mesophyll conductance according to Harley et al. 1992. It also tests the reliability of the calculation and calculates a mean with only reliable values. Note that the output is in units of umol m-2 s-1 Pa-1.

References


Examples

```r
# Read in your data
# Note that this data is coming from data supplied by the package
# hence the complicated argument in read.csv()
```
# This dataset is a CO2 by light response curve for a single sunflower

```r
data <- read.csv(system.file("extdata", "A_Ci_Q_data_1.csv", 
   package = "photosynthesis")
)
```

# Note: there will be issues here if the alpha value used  
# for calculating ETR is off, if gamma_star is incorrect,  
# if R_d is incorrect.
```r
data <- fit_g_mc_variableJ(data,  
   varnames = list(
      A_net = "A",  
      J_etr = "ETR",  
      C_i = "Ci",  
      PPFD = "Qin",  
      phi_PSII = "PhiPS2"  
   ),  
   gamma_star = 46,  
   R_d = 0.153,  
   usealpha_Q = TRUE,  
   alpha_Q = 0.84,  
   beta_Q = 0.5,  
   P = 84  
)
```

# Note that many g_mc values from this method can be unreliable
```r
ggplot(data, aes(x = CO2_s, y = g_mc, colour = reliable)) + 
   labs(
      x = expression(CO[2] ~ "(" * mu * mol ~ mol^  
         (-1) * ")" ),  
      y = expression(g[m] ~ "(mol" ~ m^(-2)  
         -1) ~ Pa^  
         (-1) * "") 
   ) + 
   geom_point(size = 2) + 
   theme_bw() + 
   theme(legend.position = "bottom")
```

# Plot QAQC graph according to Harley et al. 1992
```r
ggplot(data, aes(x = CO2_s, y = dCcdA, colour = reliable)) + 
   labs(
      x = expression(CO[2] ~ "(" * mu * mol ~ mol^  
         (-1) * ")" ),  
      y = expression(delta * C[chl] ~ "/" * delta * A)  
   ) 
```

fit_hydra_vuln_curve

fit_hydra_vuln_curve

```r
geom_hline(yintercept = 10) +
geom_point(size = 2) +
theme_bw() +
theme(legend.position = "bottom")
```

**Description**

Fitting hydraulic vulnerability curves

**Usage**

```r
fit_hydra_vuln_curve(
  data,
  varnames = list(psi = "psi", PLC = "PLC"),
  start_weibull = list(a = 2, b = 2),
  title = NULL
)
```

**Arguments**

- **data** 
  Dataframe
- **varnames** 
  List of variable names. `varnames = list(psi = "psi", PLC = "PLC")` where psi is water potential in MPa, and PLC is percent loss conductivity.
- **start_weibull** 
  starting values for the nls fitting routine for the Weibull curve
- **title** 
  Title for the output graph

**Value**

`fit_hydra_vuln_curve` fits a sigmoidal function (Pammenter & Van der Willigen, 1998) linearized according to Ogle et al. (2009). Output is a list containing the sigmoidal model in element 1 and Weibull model in element 4, the fit parameters with 95 element 2, and hydraulic parameters in element 3 (including P25, P50, P88, P95, S50, Pe, Pmax, DSI). Px (25 to 95): water potential at which x conductivity is lost. S50: slope at 50 entry point. Pmax: hydraulic failure threshold. DSI: drought stress interval. Element 5 is a graph showing the fit, P50, Pe, and Pmax.

**References**


Pammenter NW, Van der Willigen CV. 1998. A mathematical and statistical analysis of the curves illustrating vulnerability of xylem to cavitation. Tree Physiology 18:589-593
Examples

# Read in data
data <- read.csv(system.file("extdata", "hydraulic_vulnerability.csv", 
    package = "photosynthesis" 
))

# Fit hydraulic vulnerability curve
fit <- fit_hydra_vuln_curve(data[data$Tree == 4 & data$Plot == "Control", ],
    varnames = list(
        psi = "P",
        PLC = "PLC"
    ),
    title = "Control 4"
)

# Return Sigmoidal model summary
summary(fit[[1]])

# Return Weibull model summary
summary(fit[[4]])

# Return model parameters with 95\% confidence intervals
fit[[2]]

# Return hydraulic parameters
fit[[3]]

# Return graph
fit[[5]]

# Fit many curves
fits <- fit_many(
    data = data,
    varnames = list(
        psi = "P",
        PLC = "PLC"
    ),
    group = "Tree",
    funct = fit_hydra_vuln_curve
)

# To select individuals from the many fits
# Return model summary
summary(fits[[1]][[1]]) # Returns model summary

# Return sigmoidal model output
fits[[1]][[2]]

# Return hydraulic parameters
fits[[1]][[3]]
fit_many

# Return graph
fits[[1]][[5]]

# Compile parameter outputs
pars <- compile_data(
  data = fits,
  output_type = "dataframe",
  list_element = 3
)

# Compile graphs
graphs <- compile_data(
  data = fits,
  output_type = "list",
  list_element = 5
)

---

**fit_many**  
*Fitting many functions across groups*

**Description**

Fitting many functions across groups

**Usage**

```r
fit_many(data, funct, group, progress = TRUE, ...)
```

**Arguments**

- **data**  
  Dataframe
- **funct**  
  Function to fit
- **group**  
  Grouping variables
- **progress**  
  Flag. Show progress bar?
- **...**  
  Arguments for the function to fit. Use ?functionname to read the help file on available arguments for a given function.

**Value**

fit_many fits a function across every instance of a grouping variable.
Examples

# Read in your data
# Note that this data is coming from data supplied by the package
# hence the complicated argument in read.csv()
# This dataset is a CO2 by light response curve for a single sunflower
data = read.csv(system.file("extdata", "A_Ci_Q_data_1.csv",
  package = "photosynthesis"
));

# Define a grouping factor based on light intensity to split the ACi curves
data$Q_2 = as.factor(round(data$Qin, digits = 0));

# Convert leaf temperature to K
data$T_leaf = data$Tleaf + 273.15

# Fit many curves
fits = fit_many(
  data = data,
  varnames = list(
    A_net = "A",
    T_leaf = "T_leaf",
    C_i = "Ci",
    PPFD = "Qin"
  ),
  funct = fit_aci_response,
  group = "Q_2"
)

# Print the parameters
# First set of double parentheses selects an individual group value
# Second set selects an element of the sublist
fits[[3]][[1]]

# Print the graph
fits[[3]][[2]]

# Compile graphs into a list for plotting
fits_graphs = compile_data(fits, list_element = 2)

# Compile parameters into dataframe for analysis
fits_pars = compile_data(fits, output_type = "dataframe", list_element = 1)
fit_PV_curve

fit_PV_curve Fitting pressure-volume curves

Description

Fitting pressure-volume curves

Usage

fit_PV_curve(
  data,
  varnames = list(psi = "psi", mass = "mass", leaf_mass = "leaf_mass", bag_mass =
                      "bag_mass", leaf_area = "leaf_area"),
  title = NULL
)

Arguments

data Dataframe

varnames Variable names. varnames = list(psi = "psi", mass = "mass", leaf_mass = "leaf_mass",
                                      bag_mass = "bag_mass", leaf_area = "leaf_area") where psi is leaf water potential in MPa, mass is the weighed mass of the bag and leaf in g, leaf_mass is the mass of the leaf in g, bag_mass is the mass of the bag in g, and leaf_area is the area of the leaf in cm2.

title Graph title

Value

fit_PV_curve fits pressure-volume curve data to determine: SWC: saturated water content per leaf mass (g H2O g leaf dry mass ^ -1), PI_o: osmotic potential at full turgor (MPa), psi_TLP: leaf water potential at turgor loss point (TLP) (MPa), RWC_TLP: relative water content at TLP ( C_FT: relative capacitance at full turgor (MPa ^ -1), C_TLP: relative capacitance at TLP (MPa ^ -1), and C_FTStar: absolute capacitance per leaf area (g m ^ -2 MPa ^ -1). Element 1 of the output list contains the fitted parameters, element 2 contains the water-psi graph, and element 3 contains the 1/psi-100-RWC graph.

References


Examples

```r
# Read in data
data <- read.csv(system.file("extdata", "PV_curve.csv", 
   package = "photosynthesis"
))

# Fit one PV curve
fit <- fit_PV_curve(data[data$ID == "L2", ],
   varnames = list(
      psi = "psi",
      mass = "mass",
      leaf_mass = "leaf_mass",
      bag_mass = "bag_mass",
      leaf_area = "leaf_area"
   )
)

# See fitted parameters
fit[[1]]

# Plot water mass graph
fit[[2]]

# Plot PV Curve
fit[[3]]

# Fit all PV curves in a file
fits <- fit_many(data,
   group = "ID",
   funct = fit_PV_curve,
   varnames = list(
      psi = "psi",
      mass = "mass",
      leaf_mass = "leaf_mass",
      bag_mass = "bag_mass",
      leaf_area = "leaf_area"
   )
)

# See parameters
fits[[1]][[1]]

# See water mass - water potential graph
fits[[1]][[2]]

# See PV curve
fits[[1]][[3]]

# Compile parameter outputs
pars <- compile_data(
   data = fits,
)
fit_r_light_kok

    output_type = "dataframe",
    list_element = 1
)  

# Compile the water mass - water potential graphs
graphs1 <- compile_data(
    data = fits,
    output_type = "list",
    list_element = 2
)

# Compile the PV graphs
graphs2 <- compile_data(
    data = fits,
    output_type = "list",
    list_element = 3
)

---

**fit_r_light_kok**  
*Estimating light respiration*

### Description

Estimating light respiration

### Usage

```r
fit_r_light_kok(
    data,
    varnames = list(A_net = "A_net", PPFD = "PPFD"),
    PPFD_lower = 40,
    PPFD_upper = 100
)
```

```r
fit_r_light_WalkerOrt(
    data,
    varnames = list(A_net = "A_net", C_i = "C_i", PPFD = "PPFD"),
    P = 100,
    C_i_threshold = 300
)
```

```r
fit_r_light_yin(
    data,
    varnames = list(A_net = "A_net", PPFD = "PPFD", phi_PSII = "phi_PSII"),
    PPFD_lower = 40,
    PPFD_upper = 100
)
```
Arguments

data: Dataframe

varnames: List of variable names

PPFD_lower: Lower light intensity limit for estimating Rlight (Kok & Yin)

PPFD_upper: Upper light intensity limit for estimating Rlight (Kok & Yin)

P: Atmospheric pressure in kPa (Walker & Ort, 2015)

C_i_threshold: Threshold C_i (in umol mol-1) to cut data to linear region for fitting light respiration and gamma_star (Walker & Ort, 2015)

Value

fit_r_light_kok estimates light respiration using the Kok method (Kok, 1956). The Kok method involves looking for a breakpoint in the light response of net CO2 assimilation at very low light intensities and extrapolating from data above the breakpoint to estimate light respiration as the y-intercept. r_light value should be negative, denoting an efflux of CO2.

fit_r_light_WalkerOrt estimates light respiration and GammaStar according to Walk & Ort (2015) using a slope-intercept regression method to find the intercept of multiple ACi curves run at multiple light intensities. Output GammaStar and respiration should be negative. If output respiration is positive this could indicate issues (i.e., leaks) in the gas exchange measurements. GammaStar is output in umol mol-1, and respiration is output in umol m-2 s-1 of respiratory flux. Output is a list containing the slope intercept regression model, a graph of the fit, and estimates of the coefficients. NOTE: if using C_i, the output value is technically C_istar. You need to use Cc to get GammaStar. Also note, however, that the convention in the field is to completely ignore this note.

fit_r_light_yin estimates light respiration according to the Yin et al. (2009, 2011) modifications of the Kok method. The modification uses fluorescence data to get a better estimate of light respiration. Note that respiration output should be negative here to denote an efflux of CO2.

References


Walker BJ, Ort DR. 2015. Improved method for measuring the apparent CO2 photocompensation point resolves the impact of multiple internal conductances to CO2 to net gas exchange. Plant Cell Environ 38:2462-2474


Examples

# FITTING KOK METHOD
# Read in your data
# Note that this data is coming from data supplied by the package
# hence the complicated argument in read.csv()
# This dataset is a CO2 by light response curve for a single sunflower
data <- read.csv(system.file("extdata", "A_Ci_Q_data_1.csv",
package = "photosynthesis"
))

# Fit light respiration with Kok method
r_light <- fit_r_light_kok(
data = data,
varnames = list(
  A_net = "A",
  PPFD = "Qin"
),
PPFD_lower = 20,
PPFD_upper = 150
)
# Return r_light
r_light

# FITTING WALKER-ORT METHOD
# Read in your data
# Note that this data is coming from data supplied by the package
# hence the complicated argument in read.csv()
# This dataset is a CO2 by light response curve for a single sunflower
data <- read.csv(system.file("extdata", "A_Ci_Q_data_1.csv",
package = "photosynthesis"
))

# Fit the Walker-Ort method for GammaStar and light respiration
walker_ort <- fit_r_light_WalkerOrt(data,
varnames = list(
  A_net = "A",
  C_i = "Ci",
  PPFD = "Qin"
)
)
# Extract model
summary(walker_ort[[1]])
# View graph
walker_ort[[2]]
# View coefficients
walker_ort[[3]]

# FITTING THE YIN METHOD
# Read in your data
# Note that this data is coming from data supplied by the package
# hence the complicated argument in read.csv()
# This dataset is a CO2 by light response curve for a single sunflower
data <- read.csv(system.file("extdata", "A_CI_Q_data_1.csv",
  package = "photosynthesis"
))

# Fit light respiration with Yin method
r_light <- fit_r_light_yin(
  data = data,
  varnames = list(
    A_net = "A",
    PPFD = "Qin",
    phi_PSII = "PhiPS2"
  ),
  PPFD_lower = 20,
  PPFD_upper = 250
)

---

fit_t_response  Fitting temperature responses

Description

Fitting temperature responses

Usage

fit_t_response(
  data,
  varnames = list(Par = "Par", T_leaf = "T_leaf"),
  model = c("Arrhenius", "Kruse", "Heskel", "Medlyn", "MMRT", "Quadratic", "Topt"),
  start = list(a = 1, b = 1, c = 1, dEa = 1, Ea_ref = 1, Par_ref = 1, Ea = 40000, Par25 = 50, Hd = 2e+05, dS = 650, dCp = 1, dG = 1, dH = 1),
  setvar = "none",
  hdset = 2e+05,
  dSset = 650,
  title = NULL,
  ...
)

Arguments

data  Dataframe with temperature response variables

varnames  Variable names, where Par is the parameter of interest, and T_leaf is the leaf temperature in K.
Which temperature response model do you want to use? Defaults to all: Arrhenius, Heskel, Kruse, Medlyn, MMRT, Quadratic, and Topt.

List of starting parameters for the nls model fits. a, b, and c are needed for the Heskel model, dEa, Ea_ref, and Par_ref are needed for the Kruse model, Ea, Par25, and Hd are all needed for the Medlyn and Topt models while the Medlyn model also requires dS, and dCP, dG, and dH are all for the MMRT model.

Which variable to set as constant for the Medlyn model? Defaults to "none", while "Hd" and "dS" options are available.

Which value should Hd be set to when setvar = "Hd"? Specify in J/mol.

Which value should dS be set to when setvar = "dS"? Specify in J/mol/K.

Title of output graphs

Further arguments to pass on to the nlsLM() function

Value

fit_t_response fits one or more temperature response models to a dataset, returning a list of lists. The parent list contains the models, while the child list for each model contains the fitted model in element 1, the coefficients in element 2, and a graph in element 3.

References


Examples

# Read in data
data <- read.csv(system.file("extdata", "A_Ci_T_data.csv", package = "photosynthesis")
library(tidyr)

# Round temperatures to group them appropriately
# Use sequential rounding
data$T2 <- round(data$Tleaf, 1)
data$T2 <- round(data$Tleaf, 0)

# Look at unique values to detect rounding issues
unique(data$T2)

# Some still did not round correctly,
# manually correct
for (i in 1:nrow(data)) {
  if (data$T2[i] == 18) {
    data$T2[i] <- 17
  }
  if (data$T2[i] == 23) {
    data$T2[i] <- 22
  }
  if (data$T2[i] == 28) {
    data$T2[i] <- 27
  }
  if (data$T2[i] == 33) {
    data$T2[i] <- 32
  }
  if (data$T2[i] == 38) {
    data$T2[i] <- 37
  }
}

# Make sure it is a character string for grouping
data$T2 <- as.character(data$T2)

# Create grouping variable by ID and measurement temperature
data <- unite(data, 
  col = "ID2", c("ID", "T2"), 
  sep = "_")

# Split by temperature group
data <- split(data, data$ID2)

# Obtain mean temperature for group so temperature
# response fitting is acceptable later, round to
# 2 decimal places
for (i in 1:length(data)) {
  data[[i]]$Curve_Tleaf <- round(mean(data[[i]]$Tleaf), 2)
}
# Convert from list back to dataframe
data <- do.call("rbind", data)

# Parse grouping variable by ID and measurement temperature
data <- separate(data, 
col = "ID2", into = c("ID", "T2"), 
sep = "_"
)

# Make sure number of values matches number of measurement temperatures. May vary slightly if plants had slightly different leaf temperatures during the measurements
unique(data$Curve_Tleaf)

# Create ID column to curve fit by ID and temperature
data <- unite(data, 
col = "ID2", c("ID", "Curve_Tleaf"), 
sep = "_"
)

# Convert leaf temperature to K
data$T_leaf <- data$Tleaf + 273.15

# Fit many CO2 response curves
fits2 <- fit_many(
data = data, 
group = "ID2", 
varnames = list( 
  A_net = "A", 
  C_i = "Ci", 
  T_leaf = "T_leaf", 
  PPFD = "Qin", 
  g_mc = "g_mc"
), 
funct = fit_aci_response, 
alphag = 0 
)

# Extract ACi parameters
pars <- compile_data(fits2, 
  output_type = "dataframe", 
  list_element = 1 
)

# Extract ACi graphs
graphs <- compile_data(fits2, 
  output_type = "list", 
  list_element = 2 
)

# Parse the ID variable
pars <- separate(pars, col = "ID", into = c("ID", "Curve_Tleaf"), sep = ")")
# Make sure curve leaf temperature is numeric
pars$Curve_Tleaf <- as.numeric(pars$Curve_Tleaf)
pars$T_leaf <- pars$Curve_Tleaf + 273.15

# Fit all models, set Hd to constant in Medlyn model
out <- fit_t_response(
  data = pars[pars$ID == "S2", ],
  varnames = list(
    Par = "V_cmax",
    T_leaf = "T_leaf"
  ),
  setvar = "Hd",
  hdset = 200000
)

out[["Arrhenius"]][["Graph"]]
out[["Heskel"]][["Graph"]]
out[["Kruse"]][["Graph"]]
out[["Medlyn"]][["Graph"]]
out[["MMRT"]][["Graph"]]
out[["Quadratic"]][["Graph"]]
out[["Topt"]][["Graph"]]

---

**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:

\[ W_{\text{carbox}} = V_{c,max}C_{\text{chl}}/(C_{\text{chl}} + K_m) \]

where:

\[ K_m = K_C(1 + O/K_O) \]

RuBP regeneration-limited assimilation rate:

\[ W_{\text{regen}} = J C_{\text{chl}}/(4C_{\text{chl}} + 8\Gamma*) \]

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

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

TPU-limited assimilation rate:

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

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
<th>Units</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>( C_{\text{chl}} )</td>
<td>chloroplastic CO2 concentration</td>
<td>Pa</td>
<td>input</td>
</tr>
<tr>
<td>( \Gamma* )</td>
<td>chloroplastic CO2 compensation point (T_leaf)</td>
<td>Pa</td>
<td>calculated</td>
</tr>
<tr>
<td>( J_{\text{max}} )</td>
<td>potential electron transport (T_leaf)</td>
<td>( \mu \text{mol CO2} / (\text{m}^2 \text{s}) )</td>
<td>calculated</td>
</tr>
<tr>
<td>( K_C )</td>
<td>Michaelis constant for carboxylation (T_leaf)</td>
<td>( \mu \text{mol} / \text{mol} )</td>
<td>calculated</td>
</tr>
<tr>
<td>( K_O )</td>
<td>Michaelis constant for oxygenation (T_leaf)</td>
<td>( \mu \text{mol} / \text{mol} )</td>
<td>calculated</td>
</tr>
<tr>
<td>( O )</td>
<td>atmospheric O2 concentration</td>
<td>kPa</td>
<td>21.27565</td>
</tr>
<tr>
<td>( \phi_J )</td>
<td>initial slope of the response of J to PPFD</td>
<td>none</td>
<td>0.331</td>
</tr>
<tr>
<td>PPFD</td>
<td>photosynthetic photon flux density</td>
<td>umol quanta / (m^2 s)</td>
<td>1500</td>
</tr>
<tr>
<td>( R_d )</td>
<td>nonphotorespiratory CO2 release (T_leaf)</td>
<td>( \mu \text{mol CO2} / (\text{m}^2 \text{s}) )</td>
<td>calculated</td>
</tr>
<tr>
<td>( \theta_J )</td>
<td>curvature factor for light-response curve</td>
<td>none</td>
<td>0.825</td>
</tr>
<tr>
<td>( V_{c,max} )</td>
<td>maximum rate of carboxylation (T_leaf)</td>
<td>( \mu \text{mol CO2} / (\text{m}^2 \text{s}) )</td>
<td>calculated</td>
</tr>
<tr>
<td>( V_{\text{tpu}} )</td>
<td>rate of triose phosphate utilization (T_leaf)</td>
<td>( \mu \text{mol CO2} / (\text{m}^2 \text{s}) )</td>
<td>calculated</td>
</tr>
</tbody>
</table>
Value

A list of four values with units \( \text{umol CO}_2 / (\text{m}^2 \text{ s}) \) of class units:
- \( W_{\text{carbox}} \): Rubisco-limited assimilation rate
- \( W_{\text{regen}} \): RuBP regeneration-limited assimilation rate
- \( W_{\text{tpu}} \): TPU-limited assimilation rate
- \( A \): minimum of \( W_{\text{carbox}} \), \( W_{\text{regen}} \), and \( W_{\text{tpu}} \)

References


Examples

```r
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, enviro_par, bake_par, constants)

pars = c(leaf_par, enviro_par, constants)
C_chl = set_units(246.0161, umol / mol)
FvCB(C_chl, pars)
```

---

**gs_mod_ballberry**  
Stomatal conductance models

Description

Stomatal conductance models

Usage

```r
gs_mod_ballberry(A_net, C_air, RH)
gs_mod_leuning(A_net, C_air, D0, VPD)
gs_mod_opti(g0, g1, VPD, A_net, C_air)
gs_mod_optifull(g0, g1, gk, VPD, A_net, C_air)
```
Arguments

\begin{itemize}
\item \texttt{A\_net} \quad \text{Net CO2 assimilation in \textit{umol m}^{-2} \textit{s}^{-1}} \\
\item \texttt{C\_air} \quad \text{CO2 concentration at the leaf surface in \textit{umol mol}^{-1}} \\
\item \texttt{RH} \quad \text{Relative humidity as a proportion} \\
\item \texttt{D0} \quad \text{Vapor pressure sensitivity of stomata (Leuning 1995)} \\
\item \texttt{VPD} \quad \text{Vapor pressure deficit (kPa)} \\
\item \texttt{g0} \quad \text{Optimization model intercept term (Medlyn et al. 2011)} \\
\item \texttt{g1} \quad \text{Optimization model slope term (Medlyn et al. 2011)} \\
\item \texttt{gk} \quad \text{Optimization model root term (Medlyn et al. 2011)}
\end{itemize}

Value

- \texttt{gs\_mod\_ballberry} is used for fitting the Ball et al. (1987) model of stomatal conductance
- \texttt{gs\_mod\_leuning} is used for fitting the Leuning (1995) model of stomatal conductance
- \texttt{gs\_mod\_opti} fits the optimal stomatal conductance model according to Medlyn et al. 2011
- \texttt{gs\_mod\_optifull} fits the full optimal stomatal conductance model according to Medlyn et al. 2011

References


\[ J \]
\textit{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

\[ J(\texttt{pars}, \texttt{unitless} = \texttt{FALSE}) \]

Arguments

\begin{itemize}
\item \texttt{pars} \quad \text{Concatenated parameters (leaf\_par, enviro\_par, and constants)} \\
\item \texttt{unitless} \quad \text{Logical. Should units be set? The function is faster when \texttt{FALSE}, but input must be in correct units or else results will be incorrect without any warning.} 
\end{itemize}
Details

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

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

<table>
<thead>
<tr>
<th>Symbol</th>
<th>R</th>
<th>Description</th>
<th>Units</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>$J_{\text{max}}$</td>
<td>$J_{\text{max}}$</td>
<td>potential electron transport (T_leaf)</td>
<td>$\mu$mol CO$_2$ / (m$^2$s)</td>
<td>calculated</td>
</tr>
<tr>
<td>$\phi_J$</td>
<td>phi_J</td>
<td>initial slope of the response of $J$ to PPFD</td>
<td>none</td>
<td>0.331</td>
</tr>
<tr>
<td>PPFD</td>
<td>PPFD</td>
<td>photosynthetic photon flux density</td>
<td>$\mu$mol quanta / (m$^2$s)</td>
<td>1500</td>
</tr>
<tr>
<td>$\theta_J$</td>
<td>theta_J</td>
<td>curvature factor for light-response curve</td>
<td>none</td>
<td>0.825</td>
</tr>
</tbody>
</table>

Value

Value in $\mu$mol / (m$^2$s) of class units

Examples

```r
library(magrittr)
library(photosynthesis)

bake_par = make_bakepar()
constants = make_constants(use_tealeaves = FALSE)
envio_par = make_enviropar(use_tealeaves = FALSE)
leaf_par = make_leafpar(use_tealeaves = FALSE)
envio_par$T_{\text{air}} = leaf_par$T_{\text{leaf}}
leaf_par %<>% bake(envio_par, bake_par, constants)

pars = c(leaf_par, envio_par, constants)
J(pars, FALSE)
```

leaf_par  

S3 class leaf_par

Description

S3 class leaf_par

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`.
Value

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

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:

<table>
<thead>
<tr>
<th>Symbol</th>
<th>R</th>
<th>Description</th>
<th>Units</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>$D_c,0$</td>
<td>$D_{c0}$</td>
<td>diffusion coefficient for CO2 in air at 0 °C</td>
<td>m$^2$/s</td>
<td>$1.29 \times 10^{-5}$</td>
</tr>
<tr>
<td>$D_h,0$</td>
<td>$D_{h0}$</td>
<td>diffusion coefficient for heat in air at 0 °C</td>
<td>m$^2$/s</td>
<td>$1.90 \times 10^{-5}$</td>
</tr>
<tr>
<td>$D_m,0$</td>
<td>$D_{m0}$</td>
<td>diffusion coefficient for momentum in air at 0 °C</td>
<td>m$^2$/s</td>
<td>$1.33 \times 10^{-5}$</td>
</tr>
<tr>
<td>$D_w,0$</td>
<td>$D_{w0}$</td>
<td>diffusion coefficient for water vapor in air at 0 °C</td>
<td>m$^2$/s</td>
<td>$2.12 \times 10^{-5}$</td>
</tr>
<tr>
<td>$\epsilon$</td>
<td>epsilon</td>
<td>ratio of water to air molar masses</td>
<td>none</td>
<td>0.622</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>$eT$</td>
<td>eT</td>
<td>exponent for temperature dependence of diffusion</td>
<td>none</td>
<td>1.75</td>
</tr>
<tr>
<td>$R$</td>
<td>R</td>
<td>ideal gas constant</td>
<td>J / mol / K</td>
<td>8.31</td>
</tr>
<tr>
<td>$\sigma$</td>
<td>sigma</td>
<td>Stephan-Boltzmann constant</td>
<td>W / m$^2$/ K$^4$</td>
<td>$5.67 \times 10^{-8}$</td>
</tr>
</tbody>
</table>
\[ f_{Sh} \quad f_{Sh} \quad \text{function to calculate constant(s) for Sherwood number} \quad \text{none} \quad \text{NA} \]
\[ f_{Nu} \quad f_{Nu} \quad \text{function to calculate constant(s) for Nusselt number} \quad \text{none} \quad \text{NA} \]

**Baking (i.e. temperature response) parameters:**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
<th>Units</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>( D_{s,gmc} )</td>
<td>empirical temperature response parameter</td>
<td>J / mol / K</td>
<td>487</td>
</tr>
<tr>
<td>( D_{s,Jmax} )</td>
<td>empirical temperature response parameter</td>
<td>J / mol / K</td>
<td>388</td>
</tr>
<tr>
<td>( E_{a,gmc} )</td>
<td>empirical temperature response parameter</td>
<td>J / mol</td>
<td>24500</td>
</tr>
<tr>
<td>( E_{a,Jmax} )</td>
<td>empirical temperature response parameter</td>
<td>J / mol</td>
<td>68900</td>
</tr>
<tr>
<td>( E_{a,KC} )</td>
<td>empirical temperature response parameter</td>
<td>J / mol</td>
<td>56100</td>
</tr>
<tr>
<td>( E_{a,KO} )</td>
<td>empirical temperature response parameter</td>
<td>J / mol</td>
<td>81000</td>
</tr>
<tr>
<td>( E_{a,Rd} )</td>
<td>empirical temperature response parameter</td>
<td>J / mol</td>
<td>23700</td>
</tr>
<tr>
<td>( E_{a,Max} )</td>
<td>empirical temperature response parameter</td>
<td>J / mol</td>
<td>40400</td>
</tr>
<tr>
<td>( E_{a,Vcmax} )</td>
<td>empirical temperature response parameter</td>
<td>J / mol</td>
<td>52200</td>
</tr>
<tr>
<td>( E_{a,Vtpu} )</td>
<td>empirical temperature response parameter</td>
<td>J / mol</td>
<td>149000</td>
</tr>
<tr>
<td>( E_{d,gmc} )</td>
<td>empirical temperature response parameter</td>
<td>J / mol</td>
<td>121000</td>
</tr>
</tbody>
</table>

**Environment parameters:**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
<th>Units</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>( C_{air} )</td>
<td>atmospheric CO2 concentration</td>
<td>umol/mol</td>
<td>420</td>
</tr>
<tr>
<td>( O )</td>
<td>atmospheric O2 concentration</td>
<td>mol/mol</td>
<td>0.21</td>
</tr>
<tr>
<td>( P )</td>
<td>atmospheric pressure</td>
<td>kPa</td>
<td>101</td>
</tr>
<tr>
<td>PPFD</td>
<td>photosynthetic photon flux density</td>
<td>umol / m² / s</td>
<td>1500</td>
</tr>
<tr>
<td>RH</td>
<td>relative humidity</td>
<td>none</td>
<td>0.5</td>
</tr>
<tr>
<td>( u )</td>
<td>windspeed</td>
<td>m / s</td>
<td>2</td>
</tr>
</tbody>
</table>

**Leaf parameters:**

<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</td>
<td>m</td>
<td>0.1</td>
</tr>
<tr>
<td>( \Gamma^{25} )</td>
<td>chloroplastic CO2 compensation point (T_leaf)</td>
<td>umol/mol</td>
<td>NA</td>
</tr>
<tr>
<td>( g_{mc} )</td>
<td>mesophyll conductance to CO2 (T_leaf)</td>
<td>mol / m² / s</td>
<td>37.9</td>
</tr>
<tr>
<td>( g_{sc} )</td>
<td>stomatal conductance to CO2</td>
<td>mol / m² / s</td>
<td>0.4</td>
</tr>
<tr>
<td>( g_{uc} )</td>
<td>cuticular conductance to CO2</td>
<td>mol / m² / s</td>
<td>0.01</td>
</tr>
<tr>
<td>( J_{max,25} )</td>
<td>potential electron transport (25 °C)</td>
<td>umol/m²/s</td>
<td>200</td>
</tr>
<tr>
<td>( J_{max} )</td>
<td>potential electron transport (T_leaf)</td>
<td>umol/m²/s</td>
<td>NA</td>
</tr>
<tr>
<td>( k_{mc} )</td>
<td>partition of g_mc to lower mesophyll</td>
<td>none</td>
<td>1</td>
</tr>
<tr>
<td>( k_{sc} )</td>
<td>partition of g_sc to lower surface</td>
<td>none</td>
<td>1</td>
</tr>
<tr>
<td>( k_{uc} )</td>
<td>partition of g_uc to lower surface</td>
<td>none</td>
<td>1</td>
</tr>
<tr>
<td>( K_{C,25} )</td>
<td>Michaelis constant for carboxylation (25 °C)</td>
<td>umol / mol</td>
<td>268</td>
</tr>
</tbody>
</table>
**make_parameters**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>R</th>
<th>Description</th>
<th>Units</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>$K_C$</td>
<td>$K_C$</td>
<td>Michaelis constant for carboxylation ($T_{\text{leaf}}$)</td>
<td>umol / mol</td>
<td>NA</td>
</tr>
<tr>
<td>$K_{O,25}$</td>
<td>$K_{O,25}$</td>
<td>Michaelis constant for oxygenation ($25 , ^\circ\text{C}$)</td>
<td>umol / mol</td>
<td>165000</td>
</tr>
<tr>
<td>$K_O$</td>
<td>$K_O$</td>
<td>Michaelis constant for oxygenation ($T_{\text{leaf}}$)</td>
<td>umol / mol</td>
<td>NA</td>
</tr>
<tr>
<td>$\phi_J$</td>
<td>$\phi_J$</td>
<td>initial slope of the response of $J$ to PPFD</td>
<td>none</td>
<td>0.331</td>
</tr>
<tr>
<td>$R_{d,25}$</td>
<td>$R_{d,25}$</td>
<td>nonphotorespiratory CO2 release ($25 , ^\circ\text{C}$)</td>
<td>umol / m² / s</td>
<td>2</td>
</tr>
<tr>
<td>$R_d$</td>
<td>$R_d$</td>
<td>nonphotorespiratory CO2 release ($T_{\text{leaf}}$)</td>
<td>umol / m² / s</td>
<td>NA</td>
</tr>
<tr>
<td>$\theta_J$</td>
<td>$\theta_J$</td>
<td>curvature factor for light-response curve</td>
<td>none</td>
<td>0.825</td>
</tr>
<tr>
<td>$T_{\text{leaf}}$</td>
<td>$T_{\text{leaf}}$</td>
<td>leaf temperature</td>
<td>K</td>
<td>298</td>
</tr>
<tr>
<td>$V_{c,max,25}$</td>
<td>$V_{c,max,25}$</td>
<td>maximum rate of carboxylation ($25 , ^\circ\text{C}$)</td>
<td>umol / m² / s</td>
<td>150</td>
</tr>
<tr>
<td>$V_{c,max}$</td>
<td>$V_{c,max}$</td>
<td>maximum rate of carboxylation ($T_{\text{leaf}}$)</td>
<td>umol / m² / s</td>
<td>NA</td>
</tr>
<tr>
<td>$V_{tpu,25}$</td>
<td>$V_{tpu,25}$</td>
<td>rate of triose phosphate utilization ($25 , ^\circ\text{C}$)</td>
<td>umol / m² / s</td>
<td>200</td>
</tr>
<tr>
<td>$V_{tpu}$</td>
<td>$V_{tpu}$</td>
<td>rate of triose phosphate utilization ($T_{\text{leaf}}$)</td>
<td>umol / m² / s</td>
<td>NA</td>
</tr>
</tbody>
</table>

If `use_tealeaves = TRUE`, additional parameters are:

**Constants:**

<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>$R_{\text{air}}$</td>
<td>$R_{\text{air}}$</td>
<td>specific gas constant for dry air</td>
<td>J / kg / K</td>
<td>287</td>
</tr>
</tbody>
</table>

**Baking (i.e. temperature response) parameters:**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>R</th>
<th>Description</th>
<th>Units</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>$E_q$</td>
<td>$E_q$</td>
<td>energy per mole quanta</td>
<td>kJ / mol</td>
<td>220</td>
</tr>
<tr>
<td>$f_{\text{PAR}}$</td>
<td>$f_{\text{PAR}}$</td>
<td>fraction of incoming shortwave radiation that is photosynthetically active radiation (PAR)</td>
<td>none</td>
<td>0.5</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>$T_{\text{air}}$</td>
<td>$T_{\text{air}}$</td>
<td>air temperature</td>
<td>K</td>
<td>298</td>
</tr>
<tr>
<td>$T_{\text{sky}}$</td>
<td>$T_{\text{sky}}$</td>
<td>sky temperature</td>
<td>K</td>
<td>NA</td>
</tr>
</tbody>
</table>

**Leaf parameters:**

<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>$\alpha_l$</td>
<td>absorbivity of longwave radiation (4 - 80 um)</td>
<td>none</td>
<td>0.97</td>
</tr>
<tr>
<td>$\alpha_s$</td>
<td>$\alpha_s$</td>
<td>absorbivity of shortwave radiation (0.3 - 4 um)</td>
<td>none</td>
<td>0.5</td>
</tr>
<tr>
<td>$g_{sw}$</td>
<td>$g_{sw}$</td>
<td>stomatal conductance to H2O</td>
<td>mol / m² / s</td>
<td>NA</td>
</tr>
<tr>
<td>$g_{uw}$</td>
<td>$g_{uw}$</td>
<td>cuticular conductance to H2O</td>
<td>mol / m² / s</td>
<td>NA</td>
</tr>
<tr>
<td>logit($sr$)</td>
<td>logit($sr$)</td>
<td>stomatal ratio (logit transformed)</td>
<td>none</td>
<td>NA</td>
</tr>
</tbody>
</table>
Optional leaf parameters:

<table>
<thead>
<tr>
<th>Symbol</th>
<th>R</th>
<th>Description</th>
<th>Units</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\delta$IAS,lower</td>
<td>delta_ias_lower</td>
<td>effective distance through lower internal airspace</td>
<td>um</td>
<td>NA</td>
</tr>
<tr>
<td>$\delta$IAS,upper</td>
<td>delta_ias_upper</td>
<td>effective distance through upper internal airspace</td>
<td>um</td>
<td>NA</td>
</tr>
<tr>
<td>$A_{mes},/,A$</td>
<td>$A_{mes,A}$</td>
<td>mesophyll surface area per unit leaf area</td>
<td>none</td>
<td>NA</td>
</tr>
<tr>
<td>$g_{liq,c,25}$</td>
<td>$g_{liqc25}$</td>
<td>liquid-phase conductance to CO2 (25 °C)</td>
<td>mol / m$^2$/s</td>
<td>NA</td>
</tr>
<tr>
<td>$g_{liq,c}$</td>
<td>$g_{liqc}$</td>
<td>liquid-phase conductance to CO2 (T_leaf)</td>
<td>mol / m$^2$/s</td>
<td>NA</td>
</tr>
<tr>
<td>$g_{ias,c,lower}$</td>
<td>$g_{iasc_lower}$</td>
<td>internal airspace conductance to CO2 in lower part of leaf (T_leaf)</td>
<td>mol / m$^2$/s</td>
<td>NA</td>
</tr>
<tr>
<td>$g_{ias,c,upper}$</td>
<td>$g_{iasc_upper}$</td>
<td>internal airspace conductance to CO2 in upper part of leaf (T_leaf)</td>
<td>mol / m$^2$/s</td>
<td>NA</td>
</tr>
</tbody>
</table>

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


Examples

```r
bake_par = make_bakepar()
constants = make_constants(use_tealeaves = FALSE)
enviropar = make_enviropar(use_tealeaves = FALSE)
leaf_par = make_leafpar(use_tealeaves = FALSE)

leaf_par = make_leafpar(
    replace = list(
        g_sc = set_units(0.3, mol / m$^2$/s),
        V_cmax25 = set_units(100, umol / m$^2$/s)
    ), use_tealeaves = FALSE
)
```

---

**Description**

Get vector of parameter names

**Usage**

```r
parameter_names(which, use_tealeaves)
```
photosynthesis

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.

Value

A character vector with parameter names associated with each type, "leaf", "enviro", "bake", or "constants".

Examples

parameter_names("leaf", use_tealeaves = FALSE)

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,
  check = TRUE,
  parallel = FALSE,
  use_legacy_version = FALSE
)

photo(
  leaf_par,
  enviro_par,
  bake_par,
  constants,
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.

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

parallel Logical. Should parallel processing be used via future_map?

use_legacy_version Logical. Should legacy model (<2.1.0) be used? See NEWS for further information. Default is FALSE.

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

### Inputs:

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>(D_{c,0})</td>
<td>diffusion coefficient for CO2 in air at 0 °C</td>
<td>m² / s</td>
</tr>
<tr>
<td>(D_{h,0})</td>
<td>diffusion coefficient for heat in air at 0 °C</td>
<td>m² / s</td>
</tr>
<tr>
<td>(D_{m,0})</td>
<td>diffusion coefficient for momentum in air at 0 °C</td>
<td>m² / s</td>
</tr>
<tr>
<td>(D_{w,0})</td>
<td>diffusion coefficient for water vapor in air at 0 °C</td>
<td>m² / s</td>
</tr>
<tr>
<td>(\epsilon)</td>
<td>ratio of water to air molar masses</td>
<td>none</td>
</tr>
<tr>
<td>(G)</td>
<td>gravitational acceleration</td>
<td>m / s²</td>
</tr>
<tr>
<td>(\epsilon T)</td>
<td>exponent for temperature dependence of diffusion</td>
<td>none</td>
</tr>
<tr>
<td>(R)</td>
<td>ideal gas constant</td>
<td>J / mol / K</td>
</tr>
<tr>
<td>(\sigma)</td>
<td>Stephan-Boltzmann constant</td>
<td>W / m²</td>
</tr>
<tr>
<td>(f_{sh})</td>
<td>function to calculate constant(s) for Sherwood number</td>
<td>none</td>
</tr>
<tr>
<td>(f_{nu})</td>
<td>function to calculate constant(s) for Nusselt number</td>
<td>none</td>
</tr>
<tr>
<td>(D_{g,\text{gmc}})</td>
<td>empirical temperature response parameter</td>
<td>J / mol / K</td>
</tr>
<tr>
<td>(D_{s,\text{Jmax}})</td>
<td>empirical temperature response parameter</td>
<td>J / mol / K</td>
</tr>
<tr>
<td>(E_{a,\text{gmc}})</td>
<td>empirical temperature response parameter</td>
<td>J / mol / K</td>
</tr>
<tr>
<td>(E_{a,\text{Jmax}})</td>
<td>empirical temperature response parameter</td>
<td>J / mol / K</td>
</tr>
<tr>
<td>(E_{a,\text{KO}})</td>
<td>empirical temperature response parameter</td>
<td>J / mol / K</td>
</tr>
<tr>
<td>(E_{a,\text{Rd}})</td>
<td>empirical temperature response parameter</td>
<td>J / mol / K</td>
</tr>
<tr>
<td>(E_{a,\text{Vcmax}})</td>
<td>empirical temperature response parameter</td>
<td>J / mol / K</td>
</tr>
<tr>
<td>(E_{a,\text{Vtpu}})</td>
<td>empirical temperature response parameter</td>
<td>J / mol / K</td>
</tr>
<tr>
<td>(E_{d,\text{gmc}})</td>
<td>empirical temperature response parameter</td>
<td>J / mol / K</td>
</tr>
<tr>
<td>(E_{d,\text{Jmax}})</td>
<td>empirical temperature response parameter</td>
<td>J / mol / K</td>
</tr>
<tr>
<td>(C_{air})</td>
<td>atmospheric CO2 concentration</td>
<td>umol/mol</td>
</tr>
<tr>
<td>(O)</td>
<td>atmospheric O2 concentration</td>
<td>mol/mol</td>
</tr>
<tr>
<td>(P)</td>
<td>atmospheric pressure</td>
<td>kPa</td>
</tr>
<tr>
<td>PPFD</td>
<td>photosynthetic photon flux density</td>
<td>umol/m² / s</td>
</tr>
<tr>
<td>RH</td>
<td>relative humidity</td>
<td>none</td>
</tr>
<tr>
<td>(u)</td>
<td>windspeed</td>
<td>m / s</td>
</tr>
<tr>
<td>(d)</td>
<td>leaf characteristic dimension</td>
<td>m</td>
</tr>
<tr>
<td>(\Gamma^{*}_{25})</td>
<td>chloroplastic CO2 compensation point (25 °C)</td>
<td>umol/m²</td>
</tr>
<tr>
<td>(g_{mc,25})</td>
<td>mesophyll conductance to CO2 (25 °C)</td>
<td>mol / m² / s</td>
</tr>
<tr>
<td>(g_{sc})</td>
<td>stomatal conductance to CO2</td>
<td>mol / m² / s</td>
</tr>
<tr>
<td>(g_{uc})</td>
<td>cuticular conductance to CO2</td>
<td>mol / m² / s</td>
</tr>
<tr>
<td>(J_{\text{max,25}})</td>
<td>potential electron transport (25 °C)</td>
<td>umol / m² / s</td>
</tr>
<tr>
<td>(k_{mc})</td>
<td>partition of g_mc to lower mesophyll</td>
<td>none</td>
</tr>
<tr>
<td>(k_{sc})</td>
<td>partition of g_sc to lower surface</td>
<td>none</td>
</tr>
<tr>
<td>(k_{uc})</td>
<td>partition of g_uc to lower surface</td>
<td>none</td>
</tr>
<tr>
<td>(K_{C,25})</td>
<td>Michaelis constant for carboxylation (25 °C)</td>
<td>umol / m²</td>
</tr>
<tr>
<td>(K_{O,25})</td>
<td>Michaelis constant for oxygenation (25 °C)</td>
<td>umol / m²</td>
</tr>
<tr>
<td>(\phi_J)</td>
<td>initial slope of the response of J to PPFD</td>
<td>none</td>
</tr>
<tr>
<td>(R_{d,25})</td>
<td>nonphotorespiratory CO2 release (25 °C)</td>
<td>umol / m²</td>
</tr>
</tbody>
</table>
\[ \theta_J \]
\[
T_{\text{leaf}} \quad T_{\text{leaf}}
\]
\[
V_{c,\text{max},25} \quad V_{c\text{max}25}
\]
\[
V_{\text{tpu},25} \quad V_{\text{tpu}25}
\]
\[
\delta_{\text{ias,lower}} \quad \delta_{\text{ias,lower}}
\]
\[
\delta_{\text{ias,upper}} \quad \delta_{\text{ias,upper}}
\]
\[
A_{\text{mes}}/A \quad A_{\text{mes}}/A
\]
\[
g_{\text{liq,c},25} \quad g_{\text{liq}c25}
\]

**Baked Inputs:**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>R</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\Gamma^*)</td>
<td>gamma_star</td>
<td>chloroplast CO2 compensation point ((T_{\text{leaf}}))</td>
</tr>
<tr>
<td>(g_{mc})</td>
<td>g_mc</td>
<td>mesophyll conductance to CO2 ((T_{\text{leaf}}))</td>
</tr>
<tr>
<td>(J_{\text{max}})</td>
<td>j_max</td>
<td>potential electron transport ((T_{\text{leaf}}))</td>
</tr>
<tr>
<td>(K_C)</td>
<td>K_C</td>
<td>Michaelis constant for carboxylation ((T_{\text{leaf}}))</td>
</tr>
<tr>
<td>(K_O)</td>
<td>K_O</td>
<td>Michaelis constant for oxygenation ((T_{\text{leaf}}))</td>
</tr>
<tr>
<td>(R_d)</td>
<td>R_d</td>
<td>nonphotorespiratory CO2 release ((T_{\text{leaf}}))</td>
</tr>
<tr>
<td>(V_{c,\text{max}})</td>
<td>V_cmax</td>
<td>maximum rate of carboxylation ((T_{\text{leaf}}))</td>
</tr>
<tr>
<td>(V_{\text{tpu}})</td>
<td>V_tpu</td>
<td>rate of triose phosphate utilization ((T_{\text{leaf}}))</td>
</tr>
<tr>
<td>(g_{\text{liq,c}})</td>
<td>g_liqc</td>
<td>liquid-phase conductance to CO2 ((T_{\text{leaf}}))</td>
</tr>
<tr>
<td>(g_{\text{ias,c,lower}})</td>
<td>g_iasc_lower</td>
<td>internal airspace conductance to CO2 in lower part of leaf ((T_{\text{leaf}}))</td>
</tr>
<tr>
<td>(g_{\text{ias,c,upper}})</td>
<td>g_iasc_upper</td>
<td>internal airspace conductance to CO2 in upper part of leaf ((T_{\text{leaf}}))</td>
</tr>
</tbody>
</table>

**Output:**

- \(A\) photosynthetic rate at \(C_{\text{chl}}\) (\(\mu\text{mol CO}_2/ (\text{m}^2 \text{s})\))
- \(C_{\text{chl}}\) chloroplastic CO2 concentration where \(A_{\text{supply}}\) intersects \(A_{\text{demand}}\) (Pa)
- \(g_{\text{tc}}\) total conductance to CO2 at \(T_{\text{leaf}}\) (\(\mu\text{mol CO}_2/ (\text{m}^2 \text{s})\))
- value \(A_{\text{supply}} - A_{\text{demand}}\) (\(\mu\text{mol} / (\text{m}^2 \text{s})\)) at \(C_{\text{chl}}\)
- convergence convergence code (0 = converged)

**Examples**

```r
# 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)
```

```r
# Multiple parameter sets with 'photosynthesis'

leaf_par = make_leafpar(
    replace = list(
        T_leaf = set_units(c(293.14, 298.15), "K")
    )
```

```r
```
photo_parameters

Input parameters to simulate C3 photosynthesis using
photosynthesis

Description
A table of input parameters used in photosynthesis

Usage

photo_parameters

Format

```
## 'photo_parameters' A data frame with 'r nrow(photo_parameters)' rows and 'r ncol(photo_parameters)' columns:

country  Country name
iso2, iso3  2 & 3 letter ISO country codes
year  Year ...
```

Source

<https://www.who.int/teams/global-tuberculosis-programme/data>

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

```python
ppm = set_units(400, "umol/mol")
P = set_units(101.325, "kPa")
ppm2pa(ppm, P)
```

---

print_graphs  Printing graphs to system

Description

Printing graphs to system

Usage

```python
print_graphs(
    data,
    path,
    output_type = "jpeg",
    height = 5,
    width = 5,
    res = 600,
    units = "in",
    pdf_filename,
    ...
)
```

Arguments

data  List of graphs
path  File path for printing our graphs. Use "./" to set to current working directory
output_type  Type of output file, jpeg or pdf
height  Height of jpegs
width  Width of jpegs
res  Resolution of jpegs
print_graphs

units Units of height and width
pdf_filename Filename for pdf option
... Further arguments for jpeg() and pdf()

Value

print_graphs creates graph files in current working directory from a list of graphs

Examples

# Read in your data
# Note that this data is coming from data supplied by the package
# hence the complicated argument in read.csv()
# This dataset is a CO2 by light response curve for a single sunflower
data <- read.csv(system.file("extdata", "A_CI_Q_data_1.csv", 
   package = "photosynthesis")
))

# Fit many AQ curves
# Set your grouping variable
# Here we are grouping by CO2_s and individual
data$C_s <- (round(data$CO2_s, digits = 0))

# For this example we need to round sequentially due to CO2_s setpoints
data$C_s <- as.factor(round(data$C_s, digits = -1))

# To fit one AQ curve
fit <- fit_aq_response(data[data$C_s == 600, ],
   varnames = list(
      A_net = "A",
      PPFD = "Qin"
   )
)

# Print model summary
summary(fit[[1]])

# Print fitted parameters
fit[[2]]

# Print graph
fit[[3]]

# Fit many curves
fits <- fit_many(
   data = data,
   varnames = list(
      A_net = "A",
      PPFD = "Qin",
      group = "C_s"
   ),
)
funct = fit_aq_response,
group = "C_s"
)

# Look at model summary for a given fit
# First set of double parentheses selects an individual group value
# Second set selects an element of the sublist
summary(fits[[3]][[1]])

# Print the parameters
fits[[3]][[2]]

# Print the graph
fits[[3]][[3]]

# Compile graphs into a list for plotting
fits_graphs <- compile_data(fits,
  list_element = 3
)

# Print graphs to pdf
# Uncomment to run
# print_graphs(data = fits_graphs,
#   output_type = "pdf",
#   path = tempdir(),
#   pdf_filename = "mygraphs.pdf")

---

read_li6800

**Description**

Reading in LiCor files

**Usage**

read_li6800(x)

**Arguments**

- **x**  
  File name

**Value**

Returns a dataframe from raw LiCor files. Current support for LiCor 6800 files only. LiCor 6400 file reading will be supported in a later version.
t_response_arrhenius  Temperature response functions

Description

Temperature response functions

Usage

```plaintext
t_response_arrhenius(T_leaf, Ea)
t_response_arrhenius_kruse(dEa, Ea_ref, Par_ref, T2)
t_response_arrhenius_medlyn(T_leaf, Ea, Hd, dS)
t_response_arrhenius_topt(T_leaf, Ea, Hd, Topt)
t_response_calc_dS(Ea, Hd, Topt)
t_response_calc_topt(Hd, dS, Ea)
t_response_heskel(T_leaf, a, b, c)
t_response_mmrt(dCp, dG, dH, T_leaf)
```

Arguments

- **T_leaf**: Leaf temperature in K
- **Ea**: Activation energy in J mol-1 (Medlyn et al. 2002)
- **dEa**: Temperature-dependent change in Ea in K^2 (Kruse et al. 2008)
- **Ea_ref**: Activation energy in J mol-1 (Kruse et al. 2008)
- **Par_ref**: Parameter at reference temperature of 25 Celsius (Kruse et al. 2008)
- **T2**: Leaf temperature term (Kruse et al. 2008)
- **Hd**: Deactivation energy in J mol-1 (Medlyn et al. 2002)
- **dS**: Entropy parameter in J mol-1 (Medlyn et al. 2002)
- **Topt**: Optimum temperature of the process in K (Medlyn et al. 2002)
- **a**: Constant to minimize residuals (Heskel et al. 2016)
- **b**: Linear coefficient to minimize residuals (Heskel et al. 2016)
- **c**: Quadratic coefficient to minimize residuals (Heskel et al. 2016)
- **dCp**: Change in heat capacity of the enzyme between the enzyme-substrate #’ and enzyme-transition states in J mol-1 K^-1 (Hobbs et al. 2013)
- **dG**: Change in Gibbs free energy of the reaction at 25 C in J mol-1 (Hobbs et al. 2013)
- **dH**: Change in enthalpy of the reaction at 25 C in J mol-1 (Hobbs et al. 2013)
Value

t_response_arrhenius calculates the rate of a process based on an Arrhenius-type curve

t_response_arrhenius_kruse fits a peaked Arrhenius response according to Kruse et al. 2008.

t_response_arrhenius_medlyn is a peaked Arrhenius response as found in Medlyn et al. 2002.

t_response_arrhenius_topt is a peaked Arrhenius temperature response function.

t_response_calc_dS calculates dS from the fitted Topt model.

t_response_calc_topt calculates Topt for a process from Arrhenius parameters.

t_response_heskel is a quadratic temperature response according to Heskel et al. 2016.

t_response_mmrt is a macromolecular rate theory temperature response according to Hobbs et al. 2013.

References


Hobbs et al. 2013. Change in heat capacity for enzyme catalysis determines temperature dependence of enzyme catalyzed rates. ACS Chemical Biology 8:2388-2393


Index

* datasets
  * photo_parameters, 57
    .get_Dx, 14
    .get_gbc, 13, 14
    .get_gbc(CO2_conductance), 11
    .get_gmc(CO2_conductance), 11
    .get_gsc(CO2_conductance), 11
    .get_gtc(CO2_conductance), 11
    .get_guc(CO2_conductance), 11
  A_demand(A_supply), 6
  A_supply, 6
  analyze_sensitivity, 3
  aq_response, 6
  bake, 8, 10, 14, 54
  bake_par, 9, 10, 52
  baked-class, 10
  calculate_j(calculate_jmax), 11
  calculate_jmax, 11
  calculated, 7, 13, 45, 48
  calculated-parameters, 10
  CO2_conductance, 11
  compile_data, 15
  compute_sensitivity, 16
  constants, 18, 52, 54
  cross_df, 54
  enviropar, 19, 52, 54
  fit_aci_response, 19
  fit_aq_response, 23
  fit_gmc_variableJ, 28
  fit_gs_model, 25
  fit_hydra_vuln_curve, 31
  fit_many, 33
  fit_PV_curve, 35
  fit_r_light_kok, 37
  fit_r_light_WalkerOrt
    (fit_r_light_kok), 37
  fit_r_light_yin(fit_r_light_kok), 37
  fit_t_response, 40
  future_map, 54
  FvCB, 44, 54
  get_f_parameter
    (calculated-parameters), 10
  gs_mod_ballberry, 46
  gs_mod_leuning(gs_mod_ballberry), 46
  gs_mod_opti(gs_mod_ballberry), 46
  gs_mod_optifull(gs_mod_ballberry), 46
  J, 47
  leaf_par, 9, 48, 52, 54
  list, 9
  make_bakepar(make_parameters), 49
  make_constants, 8
  make_constants(make_parameters), 49
  make_enviropar(make_parameters), 49
  make_leafpar(make_parameters), 49
  make_parameters, 18, 19, 48, 49, 49, 53, 54
  parameter_names, 52
  photo, 54
  photo(photosynthesis), 53
  photo_parameters, 57
  photosynthesis, 10, 53, 54, 57
  photosynthesis-package, 3
  ppm2pa, 57
  print_graphs, 58
  read_li6800, 60
  t_response_arrhenius, 61
  t_response_arrhenius_kruse
    (t_response_arrhenius), 61
  t_response_arrhenius_medlyn
    (t_response_arrhenius), 61
t_response_arrhenius_topt
    (t_response_arrhenius), 61

t_response_calc_dS
    (t_response_arrhenius), 61

t_response_calc_topt
    (t_response_arrhenius), 61

t_response_heskel
    (t_response_arrhenius), 61

t_response_mmrt (t_response_arrhenius), 61

temp_respl (bake), 8

temp_respl (bake), 8

tleaf, 18, 19, 48, 49, 53, 54

W_carbox (FvCB), 44
W_regen (FvCB), 44
W_tpu (FvCB), 44