

# Package ‘photosynthesis’

May 9, 2026

**Version** 2.1.5

**Date** 2024-11-23

**Title** Tools for Plant Ecophysiology & Modeling

**Depends** R (>= 4.0.0), ggplot2 (>= 3.4.0), minpack.lm (>= 1.2-1), units (>= 0.6.6)

**Imports** checkmate (>= 2.0.0), crayon (>= 1.3.4), dplyr (>= 0.8.5), furr (>= 0.1.0), glue (>= 1.4.0), graphics (>= 4.0.0), grDevices (>= 4.0.0), gunit (>= 1.0.2), lifecycle (>= 1.0.0), magrittr (>= 1.5.0), methods (>= 3.5.0), nlme (>= 3.1-147), progress (>= 1.2.0), purrr (>= 0.3.3), readr (>= 2.0.0), rlang (>= 0.4.6), stats (>= 4.0.0), stringr (>= 1.4.0), tealeaves (>= 1.0.5), utils (>= 4.0.0)

**Suggests** brms, broom, future, knitr, rmarkdown, testthat, tibble, tidyr, tidyselect

**Description** Contains modeling and analytical tools for plant ecophysiology. MODELING: Simulate C3 photosynthesis using the Farquhar, von Caemmerer, Berry (1980) <doi:10.1007/BF00386231> model as described in Buckley and Diaz-Espejo (2015) <doi:10.1111/pce.12459>. It uses units to ensure that parameters are properly specified and transformed before calculations. Temperature response functions get automatically ``baked" into all parameters based on leaf temperature following Bernacchi et al. (2002) <doi:10.1104/pp.008250>. The package includes boundary layer, cuticular, stomatal, and mesophyll conductances to CO<sub>2</sub>, 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 CO<sub>2</sub> conductance or photochemistry. ANALYTICAL TOOLS: Fit AC<sub>i</sub> (Farquhar et al. (1980) <doi:10.1007/BF00386231>) and AQ curves (Marshall & Biscoe (1980) <doi:10.1093/jxb/31.1.29>), temperature responses (Heskel et al. (2016) <doi:10.1073/pnas.1520282113>; Kruse et al. (2008) <doi:10.1111/j.1365-3040.2008.01809.x>, Medlyn et al. (2002) <doi:10.1046/j.1365-3040.2002.00891.x>, Hobbs et al. (2013) <doi:10.1021/cb4005029>), respiration in the light (Kok (1956) <doi:10.1016/0006-3002(56)90003-8>, Walker & Ort (2015) <doi:10.1111/pce.12562>),

Yin et al. (2009) <[doi:10.1111/j.1365-3040.2009.01934.x](https://doi.org/10.1111/j.1365-3040.2009.01934.x)>, Yin et al. (2011) <[doi:10.1093/jxb/err038](https://doi.org/10.1093/jxb/err038)>), mesophyll conductance (Harley et al. (1992) <[doi:10.1104/pp.98.4.1429](https://doi.org/10.1104/pp.98.4.1429)>), pressure-volume curves (Koide et al. (2000) <[doi:10.1007/978-94-009-2221-1\\_9](https://doi.org/10.1007/978-94-009-2221-1_9)>, Sack et al. (2003) <[doi:10.1046/j.0016-8025.2003.01058.x](https://doi.org/10.1046/j.0016-8025.2003.01058.x)>, Tyree et al. (1972) <[doi:10.1093/jxb/23.1.267](https://doi.org/10.1093/jxb/23.1.267)>), hydraulic vulnerability curves (Ogle et al. (2009) <[doi:10.1111/j.1469-8137.2008.02760.x](https://doi.org/10.1111/j.1469-8137.2008.02760.x)>, Pammenter et al. (1998) <[doi:10.1093/treephys/18.8-9.589](https://doi.org/10.1093/treephys/18.8-9.589)>), and tools for running sensitivity analyses particularly for variables with uncertainty (e.g. `g_mc()`, `gamma_star()`, `R_d()`).

**License** MIT + file LICENSE

**Encoding** UTF-8

**LazyData** true

**RoxygenNote** 7.3.2

**VignetteBuilder** knitr

**URL** <https://github.com/cdmuir/photosynthesis>

**BugReports** <https://github.com/cdmuir/photosynthesis/issues>

**NeedsCompilation** no

**Author** Joseph Stinziano [aut] (ORCID: <<https://orcid.org/0000-0002-7628-4201>>),  
Cassandra Roback [aut],  
Demi Sargent [aut],  
Bridget Murphy [aut],  
Patrick Hudson [aut, dtc],  
Chris Muir [aut, cre] (ORCID: <<https://orcid.org/0000-0003-2555-3878>>)

**Maintainer** Chris Muir <[cdmuir@wisc.edu](mailto:cdmuir@wisc.edu)>

**Repository** CRAN

**Date/Publication** 2024-11-24 19:40:02 UTC

## Contents

analyze_sensitivity . . . . .	3
aq_response . . . . .	5
A_supply . . . . .	6
bake . . . . .	7
baked-class . . . . .	9
bake_par . . . . .	10
calculated-parameters . . . . .	10
calculate_jmax . . . . .	11
CO2_conductance . . . . .	11
compile_data . . . . .	15
compute_sensitivity . . . . .	16
constants . . . . .	18
enviro_par . . . . .	19

fit_aci_response . . . . .	19
fit_aq_response . . . . .	23
fit_aq_response2 . . . . .	25
fit_gs_model . . . . .	27
fit_g_mc_variableJ . . . . .	30
fit_hydra_vuln_curve . . . . .	33
fit_many . . . . .	35
fit_photosynthesis . . . . .	37
fit_PV_curve . . . . .	38
fit_r_light2 . . . . .	40
fit_r_light_kok . . . . .	44
fit_t_response . . . . .	47
FvCB . . . . .	51
get_default_model . . . . .	53
gs_mod_ballberry . . . . .	54
J . . . . .	55
leaf_par . . . . .	56
make_parameters . . . . .	57
parameter_names . . . . .	60
photosynthesis . . . . .	61
photo_parameters . . . . .	65
ppm2pa . . . . .	65
print_graphs . . . . .	66
read_li6800 . . . . .	68
read_licor . . . . .	69
required_variables . . . . .	70
simulate_error . . . . .	70
t_response_arrhenius . . . . .	73

**Index****75**


---

analyze\_sensitivity     *Running 2-parameter sensitivity analyses*

---

**Description**

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

**[Deprecated]**

Please use `marshall_biscoe_1980()`.

**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 (umol m <sup>-2</sup> s <sup>-1</sup> )
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 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**

Marshall B, Biscoe P. 1980. A model for C3 leaves describing the dependence of net photosynthesis on irradiance. J Ex Bot 31:29-39

---

A_supply	<i>CO2 supply and demand function (mol / m<sup>2</sup> s)</i>
----------	---

---

**Description**

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

**Usage**

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

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

**Arguments**

C_chl	Chloroplastic CO2 concentration in Pa of class units
pars	Concatenated parameters (leaf_par, enviro_par, and constants)
unitless	Logical. Should units be set? The function is faster when FALSE, but input must be in correct units or else results will be incorrect without any warning.
use_legacy_version	Logical. Should legacy model (<2.1.0) be used? See <b>NEWS</b> for further information. Default is FALSE.

**Details****Supply function:**

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

**Demand function:**

$$A = (1 - \Gamma^* / C_{\text{chl}}) \min(W_{\text{carbox}}, W_{\text{regen}}, W_{\text{tpu}}) - R_d$$

<i>Symbol</i>	<i>R</i>	<i>Description</i>	<i>Units</i>	<i>Default</i>
$A$	A	photosynthetic rate	$\mu\text{mol CO}_2 / (\text{m}^2 \text{ s})$	calculated
$g_{tc}$	g_tc	total conductance to CO <sub>2</sub>	$\mu\text{mol CO}_2 / (\text{m}^2 \text{ s Pa})$	calculated
$C_{\text{air}}$	C_air	atmospheric CO <sub>2</sub> concentration	Pa	41
$C_{\text{chl}}$	C_chl	chloroplastic CO <sub>2</sub> concentration	Pa	calculated
$R_d$	R_d	nonphotorespiratory CO <sub>2</sub> release	$\mu\text{mol CO}_2 / (\text{m}^2 \text{ s})$	2
$\Gamma^*$	gamma_star	chloroplastic CO <sub>2</sub> compensation point	Pa	3.743

**Value**

Value in mol / (m<sup>2</sup> 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**

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

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

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

**Arguments**

leaf_par	A list of leaf parameters inheriting class leaf_par. This can be generated using the make_leafpar function.
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 <a href="#">make_constants()</a> .
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):

$$\text{par}(T_{\text{leaf}}) = \text{par25} \exp(E_a / (RT_{\text{ref}})) (T_{\text{leaf}} - 25) / (T_{\text{leaf}} + 273.15)$$

$T_{\text{ref}}$  is the reference temperature in K

$T_{\text{leaf}}$  is the leaf temperature in °C

Temperature response function 2 (temp\_response2) is the above equation multiplied by:

$$(1 + \exp((D_s/R - E_d/(RT_{\text{ref}})))) / (1 + \exp((D_s/R) - (E_d/(R(T_{\text{leaf}} + 273.15)))))$$

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  $T_{\text{leaf}}$  using temperature response functions detailed below.

### References

Buckley TN, Diaz-Espejo A. 2015. Partitioning changes in photosynthetic rate into contributions from different variables. *Plant, Cell and Environment* 38: 1200-1211.

### Examples

```
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	<i>Inverse non-rectangular hyperbola for J_max calculation</i>
----------------	--

---

**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 $\mu\text{mol m}^{-2} \text{s}^{-1}$
alpha	initial slope of the light response
J	electron transport rate in $\mu\text{mol m}^{-2} \text{s}^{-1}$
theta_J	curvature of the light response
J_max	maximum rate of electron transport in $\mu\text{mol m}^{-2} \text{s}^{-1}$

**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 CO<sub>2</sub> response curves in fit\_aci\_response.

---

CO2_conductance	<i>Conductance to CO2 (mol / m<sup>2</sup> / s)</i>
-----------------	---

---

**Description**

Conductance to CO<sub>2</sub> (mol / m<sup>2</sup> / s)

- g\_tc: total conductance to CO<sub>2</sub>
- g\_uc: cuticular conductance to CO<sub>2</sub>
- g\_bc: boundary layer conductance to CO<sub>2</sub>
- g\_mc: mesophyll conductance to CO<sub>2</sub>
- g\_sc: stomatal conductance to CO<sub>2</sub>

**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 <b>NEWS</b> 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:

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

The boundary layer conductances for each are calculated on the basis of mass and heat transfer (see [.get\\_gbc\(\)](#)).

Symbol	R	Description	Units	Default
$g_{mc}$	g_mc	mesophyll conductance to CO2 (T_leaf)	mol / m <sup>2</sup> / s	calculated
$g_{sc}$	g_sc	stomatal conductance to CO2	mol / m <sup>2</sup> / s	0.4
$g_{uc}$	g_uc	cuticular conductance to CO2	mol / m <sup>2</sup> / s	0.01
$k_{mc}$	k_mc	partition of $g_{mc}$ to lower mesophyll	none	1
$k_{sc}$	k_sc	partition of $g_{sc}$ to lower surface	none	1
$k_{uc}$	k_uc	partition of $g_{uc}$ to lower surface	none	1

### 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 ( $A_{mes\_A}$ ,  $g_{liqc}$ ) 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 ( $g_{liqc}$ ).

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  $r_{i,c}$ .

$$r_{total,c} = r_{gas,c} + r_{i,c}$$

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  $i \in \{lower, upper\}$  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:

$$1/r_{u+s,c,i} = g_{u+s,c,i} = g_{u,c,i} + g_{s,c,i}$$

Substituting the above expression into the equation for  $r_{\text{gas},c,i}$ :

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

The total gas-phase resistance is the inverse of the sum of the parallel lower and upper conductances:

$$1/r_{\text{gas},c} = g_{\text{gas},c,\text{lower}} + g_{\text{gas},c,\text{upper}}$$

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:

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

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

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

How the partitioning factors work:

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

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

$$g_{\text{ias},c,\text{lower}} = D_c/\delta_{\text{ias},\text{lower}}$$

$$g_{\text{ias},c,\text{upper}} = D_c/\delta_{\text{ias},\text{upper}}$$

`g_iasc_lower` and `g_iasc_upper` are calculated in the `bake` function. See `tealeaves::get_Dx()` for calculating `D_c`.

The liquid-phase intracellular resistance is given by:

$$1/r_{i,c} = g_{i,c} = g_{\text{liq},c}A_{\text{mes}}/A$$

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

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

---

compile_data	<i>Compiling outputs from lists</i>
--------------	-------------------------------------

---

## 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

```
# 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"
  )

# 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    *Computing measures of sensitivity*

---

## Description

Computing measures of sensitivity

## Usage

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

## Arguments

data	Dataframe with output from sensitivity_analysis()
varnames	Variable names
test1_ref	Reference value for parameter
test2_ref	Reference value for parameter

## 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.

## References

Bauerle WL, Daniels AB, Barnard DM. 2014. Carbon and water flux responses to physiology by environment interactions: a sensitivity analysis of variation in climate on photosynthetic and stomatal parameters. *Climate Dynamics* 42: 2539-2554.

Capaldo KP, Pandis SN 1997. Dimethylsulfide chemistry in the remote marine atmosphere: evaluation and sensitivity analysis of available mechanisms. *J Geophys Res* 102:23251-23267

## 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

<code>.x</code>	A list to be constructed into <b>constants</b> .
<code>use_tealeaves</code>	Logical. Should leaf energy balance be used to calculate leaf temperature ( <code>T_leaf</code> )? If TRUE, <code>tleaf()</code> calculates <code>T_leaf</code> . If FALSE, user-defined <code>T_leaf</code> is used. Additional parameters and constants are required, see <a href="#">make_parameters()</a> .

## Value

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

---

enviro_par	<i>S3 class enviro_par</i>
------------	----------------------------

---

**Description**

S3 class enviro\_par

**Usage**

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

**Arguments**

.x	A list to be constructed into <b>enviro_par</b> .
use_tealeaves	Logical. Should leaf energy balance be used to calculate leaf temperature (T_leaf)? If TRUE, <code>tleaf()</code> calculates T_leaf. If FALSE, user-defined T_leaf is used. Additional parameters and constants are required, see <a href="#">make_parameters()</a> .

**Value**

Constructor function for enviro\_par class. This function ensures that environmental parameter inputs are properly formatted.

---

fit_aci_response	<i>Fitting ACi curves</i>
------------------	---------------------------

---

**Description**

Fitting ACi curves

**Usage**

```
fit_aci_response(
  data,
  varnames = list(A_net = "A_net", T_leaf = "T_leaf", C_i = "C_i", PPF = "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_0 = 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_025 = NULL,
Ea_K_0 = 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", PPF = "PPFD", g_mc = "g_mc"), where A_net is net CO <sub>2</sub> assimilation, T_leaf is leaf temperature in Kelvin, C_i is intercellular CO <sub>2</sub> concentration in umol/mol, PPF is incident irradiance in umol m <sup>-2</sup> s <sup>-1</sup> (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 CO <sub>2</sub> in mol m <sup>-2</sup> s <sup>-1</sup> Pa <sup>-1</sup> .
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-CO <sub>2</sub> decline, then this value should be > 0.
R_d_meas	Measured value of respiratory CO <sub>2</sub> efflux in umol m <sup>-2</sup> s <sup>-1</sup> . 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 <i>Nicotiana tabacum</i> defaults from Bernacchi et al. 2001.
useK_M	Specify K_M? If FALSE, uses an Arrhenius temperature response function with <i>Nicotiana tabacum</i> defaults from Bernacchi et al. 2001.

useK_C_K_0	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 CO <sub>2</sub> assimilation
theta_J	Curvature of the photosynthetic light response curve
gamma_star25	gamma_star at 25C in umol mol <sup>-1</sup>
Ea_gamma_star	Activation energy of gamma_star in J mol <sup>-1</sup>
K_M25	Michaelis-Menten constant for rubisco at 25C
Ea_K_M	Activation energy for K_M in J mol <sup>-1</sup>
g_mc25	Mesophyll conductance at 25C in mol m <sup>-2</sup> s <sup>-1</sup>
Ea_g_mc	Activation energy of g_mc in J mol <sup>-1</sup>
K_C25	Michaelis-Menten constant for rubisco carboxylation at 25C
Ea_K_C	Activation energy for K_C in J mol <sup>-1</sup>
K_O25	Michaelis-Menten constant for rubisco oxygenation at 25C
Ea_K_O	Activation energy for K_O in J mol <sup>-2</sup>
Oconc	O <sub>2</sub> concentration in %. Used with P to calculate intracellular O <sub>2</sub> 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 AC<sub>i</sub> curves using an approach similar to Gu et al. 2010. Iterates all possible C<sub>i</sub> 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

- Bernacchi CJ, Singaas EL, Pimentel C, Portis AR, Long SP. 2001. Improved temperature response functions for models of rubisco-limited photosynthesis. *Plant Cell Environment* 24:253-259.
- Bernacchi CJ, Portis AR, Nakano H, von Caemmerer S, Long SP. 2002. Temperature response of mesophyll conductance. Implications for the determination of rubisco enzyme kinetics and for limitations to photosynthesis in vivo. *Plant Physiology* 130:1992-1998.
- Gu L, Pallardy SG, Tu K, Law BE, Wullschleger SD. 2010. Reliable estimation of biochemical parameters from C<sub>3</sub> leaf photosynthesis-intercellular carbon dioxide response curves. *Plant Cell Environment* 33:1852-1874.
- von Caemmerer S. 2000. *Biochemical models of leaf photosynthesis*. CSIRO Publishing, Collingwood.

**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
# 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

### [Deprecated]

Please use `fit_aq_response2()`.

## Usage

```

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

```

## Arguments

<code>data</code>	Dataframe containing CO2 assimilation light response
<code>varnames</code>	Variable names where <code>varnames = list(A_net = "A_net", PPFD = "PPFD")</code> . <code>A_net</code> is net CO2 assimilation in $\mu\text{mol m}^{-2} \text{s}^{-1}$ , <code>PPFD</code> is incident irradiance. <code>PPFD</code> can be corrected for light absorbance by using <code>usealpha_Q</code> and setting <code>alpha_Q</code> .
<code>usealpha_Q</code>	Correct light intensity for absorbance? Default is <code>FALSE</code> .
<code>alpha_Q</code>	Absorbance of incident light. Default value is 0.84.
<code>title</code>	Title for graph

## Value

fit\_aq\_response fits the light response of net CO<sub>2</sub> assimilation. Output is a dataframe containing light saturated net CO<sub>2</sub> assimilation, quantum yield of CO<sub>2</sub> assimilation (phi\_J), curvature of the light response (theta\_J), respiration (Rd), light compensation point (LCP), and residual sum of squares (resid\_SS). Note that Rd 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. Rd output should thus be interpreted more as a residual parameter to ensure an accurate fit of the light response parameters. Model originally from Marshall & Biscoe 1980.

## References

Marshall B, Biscoe P. 1980. A model for C<sub>3</sub> leaves describing the dependence of net photosynthesis on irradiance. *J Ex Bot* 31:29-39

## 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",
      PPF_D = "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\_aq\_response2

*Fit photosynthetic light-response curves*


---

### Description

We recommend using `fit_photosynthesis()` with argument `.photo_fun = "aq_response"` rather than calling this function directly.

### Usage

```

fit_aq_response2(
  .data,
  .model = "default",
  .method = "ls",
  usealpha_Q = FALSE,
  alpha_Q = 0.84,

```

```

  quiet = FALSE,
  brm_options = NULL
)

```

### Arguments

<code>.data</code>	A data frame containing plant ecophysiological data. See <code>required_variables()</code> for the variables required for each model.
<code>.model</code>	A character string of model name to use. See <code>get_all_models()</code> .
<code>.method</code>	A character string of the statistical method to use: 'ls' for least-squares and 'brms' for Bayesian model using <code>brms::brm()</code> . Default is 'ls'.
<code>usealpha_Q</code>	Flag. Should light intensity be multiplied by <code>alpha_Q</code> before fitting? Default is FALSE (i.e. assume that '.Q' is absorbed light).
<code>alpha_Q</code>	Number. Absorbance of incident light. Default value is 0.84. Ignored if <code>usealpha_Q = FALSE</code> .
<code>quiet</code>	Flag. Should messages be suppressed? Default is FALSE.
<code>brm_options</code>	A list of options passed to <code>brms::brm()</code> if <code>.method = "brms"</code> . Default is NULL.

### Value

- If `.method = 'ls'`: an `stats::nls()` object.
- If `.method = 'brms'`: a `brms::brmsfit()` object.

### Note

Rd fitted in this way is essentially the same as the Kok (1956) method, and represents a respiration value in the light that may not be accurate. Rd output should thus be interpreted more as a residual parameter to ensure an accurate fit of the light response parameters. Model originally from Marshall & Biscoe (1980).

### References

Marshall B, Biscoe P. 1980. A model for C3 leaves describing the dependence of net photosynthesis on irradiance. *J Ex Bot* 31:29-39

### Examples

```

library(broom)
library(dplyr)
library(photosynthesis)

# Read in your data
dat = system.file("extdata", "A_Ci_Q_data_1.csv", package = "photosynthesis") |>
  read.csv() |>
  # Set grouping variable
  mutate(group = round(CO2_s, digits = 0)) |>
  # For this example, round sequentially due to CO2_s set points

```

```

mutate(group = as.factor(round(group, digits = -1)))

# Fit one light-response curve
fit = fit_photosynthesis(
  .data = filter(dat, group == 600),
  .photo_fun = "aq_response",
  .vars = list(.A = A, .Q = Qabs),
)

# The 'fit' object inherits class 'nls' and many methods can be used

## Model summary:
summary(fit)

## Estimated parameters:
coef(fit)

## 95% confidence intervals:
confint(fit)

## Tidy summary table using 'broom::tidy()'
tidy(fit, conf.int = TRUE, conf.level = 0.95)

# Fit multiple curves with photosynthesis and purrr

library(purrr)

fits = dat |>
  split(~ group) |>
  map(fit_photosynthesis, .photo_fun = "aq_response", .vars = list(.A = A, .Q = Qabs))

```

---

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 contribution to the control of photosynthesis under different environmental conditions, in *Progress in Photosynthesis Research, Proceedings of the VII International Congress on Photosynthesis*, vol. 4, edited by I. Biggins, pp. 221–224, Martinus Nijhoff, Dordrecht, Netherlands.
- Leuning R. 1995. A critical appraisal of a coupled stomatal- photosynthesis model for C3 plants. *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

```
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 CO <sub>2</sub> assimilation in $\mu\text{mol m}^{-2} \text{s}^{-1}$ , J_etr is linear electron transport flux in $\mu\text{mol m}^{-2} \text{s}^{-1}$ , C_i is intercellular CO <sub>2</sub> concentration in $\mu\text{mol mol}^{-1}$ , PPFD is incident irradiance in $\mu\text{mol m}^{-2} \text{s}^{-1}$ , 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 CO <sub>2</sub> compensation point in $\mu\text{mol mol}^{-1}$
R_d	Respiration rate in $\mu\text{mol m}^{-2} \text{s}^{-1}$
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  $\mu\text{mol m}^{-2} \text{s}^{-1} \text{Pa}^{-1}$ .

### References

Harley PC, Loreto F, Di Marco G, Sharkey TD. 1992. Theoretical considerations when estimating mesophyll conductance to CO<sub>2</sub> flux by analysis of the response of photosynthesis to CO<sub>2</sub>. Plant Physiol 98:1429 - 1436.

### 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"
))

```

```

# 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.
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
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
    } ~ s^{
      -1
    } ~ Pa^
      {
        -1
      } * ")")
  ) +
  geom_point(size = 2) +
  theme_bw() +
  theme(legend.position = "bottom")

# Plot QAQC graph according to Harley et al. 1992
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)
  ) +
  geom_hline(yintercept = 10) +
  geom_point(size = 2) +
  theme_bw() +
  theme(legend.position = "bottom")

```

---

 fit\_hydra\_vuln\_curve *Fitting hydraulic vulnerability curves*


---

## Description

Fitting hydraulic vulnerability curves

## Usage

```
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% confidence interval for both models are in 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% of conductivity is lost. S50: slope at 50% loss of conductivity. Pe: air entry point. Pmax: hydraulic failure threshold. DSI: drought stress interval. Element 5 is a graph showing the fit, P50, Pe, and Pmax.

## References

Ogle K, Barber JJ, Willson C, Thompson B. 2009. Hierarchical statistical modeling of xylem vulnerability to cavitation. *New Phytologist* 182:541-554

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]]

# 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

### [Deprecated]

We are no longer updating this function. Please use generic methods like [map](#) instead. See `vignette("light-response")` for an example.

## Usage

```
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 <code>?functionname</code> 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\_photosynthesis     *Fit photosynthetic models with gas-exchange data*


---

## Description

Fit photosynthetic models with gas-exchange data

## Usage

```
fit_photosynthesis(
  .data,
  .photo_fun,
  .model = "default",
  .vars = NULL,
  .method = "ls",
  ...,
  quiet = FALSE,
  brm_options = NULL
)
```

## Arguments

<code>.data</code>	A data frame containing plant ecophysiological data. See <a href="#">required_variables()</a> for the variables required for each model.
<code>.photo_fun</code>	A character string of <b>photosynthesis</b> function to call. One of: <code>aq_response</code> , <code>r_light</code> .
<code>.model</code>	A character string of model name to use. See <a href="#">get_all_models()</a> .
<code>.vars</code>	A list to rename variables in <code>.data</code> . See <a href="#">required_variables()</a> for the accepted variable names.
<code>.method</code>	A character string of the statistical method to use: <code>'ls'</code> for least-squares and <code>'brms'</code> for Bayesian model using <code>brms::brm()</code> . Default is <code>'ls'</code> .
<code>...</code>	Additional arguments passed to specific models. See specific help pages for each type of photosynthetic model: <ul style="list-style-type: none"> <li>• Light-response curves <a href="#">fit_aq_response2()</a></li> <li>• Light respiration <a href="#">fit_r_light2()</a></li> </ul>
<code>quiet</code>	Flag. Should messages be suppressed? Default is <code>FALSE</code> .
<code>brm_options</code>	A list of options passed to <code>brms::brm()</code> if <code>.method = "brms"</code> . Default is <code>NULL</code> .

## Value

A fitted model object

- class `'lm'` or `'nls'` if `method = 'ls'`
- class `'brmsfit'` if `method = 'brms'`

**Note**

This function will fit models to data but several methods require post-processing to extract meaningful parameter estimates and confidence intervals. See vignettes for further explanation and examples.

- Light-response curves: `vignette("light-response", package = "photosynthesis")`
- Light respiration: `vignette("light-respiration", package = "photosynthesis")`

---

fit_PV_curve	<i>Fitting pressure-volume curves</i>
--------------	---------------------------------------

---

**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**

<code>data</code>	Dataframe
<code>varnames</code>	Variable names. <code>varnames = list(psi = "psi", mass = "mass", leaf_mass = "leaf_mass", bag_mass = "bag_mass", leaf_area = "leaf_area")</code> where <code>psi</code> is leaf water potential in MPa, <code>mass</code> is the weighed mass of the bag and leaf in g, <code>leaf_mass</code> is the mass of the leaf in g, <code>bag_mass</code> is the mass of the bag in g, and <code>leaf_area</code> is the area of the leaf in cm <sup>2</sup> .
<code>title</code>	Graph title

**Value**

`fit_PV_curve` fits pressure-volume curve data to determine: `SWC`: saturated water content per leaf mass ( $\text{g H}_2\text{O g leaf dry mass}^{-1}$ ), `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 (%), `eps`: modulus of elasticity at full turgor (MPa), `C_FT`: relative capacitance at full turgor ( $\text{MPa}^{-1}$ ), `C_TLP`: relative capacitance at TLP ( $\text{MPa}^{-1}$ ), and `C_FTStar`: absolute capacitance per leaf area ( $\text{g m}^{-2} \text{MPa}^{-1}$ ). 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

Koide RT, Robichaux RH, Morse SR, Smith CM. 2000. Plant water status, hydraulic resistance and capacitance. In: Plant Physiological Ecology: Field Methods and Instrumentation (eds RW Pearcy, JR Ehleringer, HA Mooney, PW Rundel), pp. 161-183. Kluwer, Dordrecht, the Netherlands

Sack L, Cowan PD, Jaikumar N, Holbrook NM. 2003. The 'hydrology' of leaves: co-ordination of structure and function in temperate woody species. *Plant, Cell and Environment*, 26, 1343-1356

Tyree MT, Hammel HT. 1972. Measurement of turgor pressure and water relations of plants by pressure bomb technique. *Journal of Experimental Botany*, 23, 267

## Examples

```
# 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,
  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\_light2

*Fit models to estimate light respiration ( $R_d$ )*


---

### Description

We recommend using `fit_photosynthesis()` with argument `.photo_fun = "r_light"` rather than calling this function directly.

### Usage

```

fit_r_light2(
  .data,
  .model = "default",
  .method = "ls",
  Q_lower = NA,
  Q_upper = NA,
  Q_levels = NULL,
  C_upper = NA,
  quiet = FALSE,
  brm_options = NULL
)

```

**Arguments**

<code>.data</code>	A data frame containing plant ecophysiological data. See <code>required_variables()</code> for the variables required for each model.
<code>.model</code>	A character string of model name to use. See <code>get_all_models()</code> .
<code>.method</code>	A character string of the statistical method to use: 'ls' for least-squares and 'brms' for Bayesian model using <code>brms::brm()</code> . Default is 'ls'.
<code>Q_lower</code>	Lower light intensity limit for estimating $R_d$ using kok_1956 and yin_etal_2011 models.
<code>Q_upper</code>	Upper light intensity limit for estimating $R_d$ using kok_1956 and yin_etal_2011 models
<code>Q_levels</code>	A numeric vector of light intensity levels ( $\mu\text{mol} / \text{mol}$ ) for estimating $R_d$ from the linear region of the A-C curve using the walker_ort_2015 model.
<code>C_upper</code>	Upper C ( $\mu\text{mol} / \text{mol}$ ) limit for estimating $R_d$ from the linear region of the A-C curve using the walker_ort_2015 model.
<code>quiet</code>	Flag. Should messages be suppressed? Default is FALSE.
<code>brm_options</code>	A list of options passed to <code>brms::brm()</code> if <code>.method = "brms"</code> . Default is NULL.

**Value**

- If `.method = 'ls'`: an `stats::nls()` or `stats::lm()` object.
- If `.method = 'brms'`: a `brms::brmsfit()` object.

**Note**

Confusingly,  $R_d$  typically denotes respiration in the light, but you might see  $R_{\text{day}}$  or  $R_{\text{light}}$ .

**Models***Kok (1956)*

The kok\_1956 model estimates light respiration using the Kok method (Kok, 1956). The Kok method involves looking for a breakpoint in the light response of net CO<sub>2</sub> assimilation at very low light intensities and extrapolating from data above the breakpoint to estimate light respiration as the y-intercept.  $R_d$  value should be negative, denoting an efflux of CO<sub>2</sub>.

*Yin et al. (2011)*

The yin\_etal\_2011 model 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.  $R_d$  values should be negative here to denote an efflux of CO<sub>2</sub>.

*Walker & Ort (2015)*

The walker\_ort\_2015 model estimates light respiration and  $\Gamma^*$  according to Walker & Ort (2015) using a slope- intercept regression method to find the intercept of multiple A-C curves run at multiple light intensities. The method estimates  $\Gamma^*$  and  $R_d$ . If estimated  $R_d$  is positive this could indicate issues (i.e. leaks) in the gas exchange measurements.  $\Gamma^*$  is in units of  $\mu\text{mol} / \text{mol}$  and  $R_d$  is in units of  $\mu\text{mol m}^{-2} \text{s}^{-1}$  of respiratory flux. If using  $C_i$ , the estimated value is technically  $C_i^*$ . You need to use  $C_c$  to get  $\Gamma^*$ . Also note, however, that the convention in the field is to completely ignore this note.

## References

Kok B. 1956. On the inhibition of photosynthesis by intense light. *Biochimica et Biophysica Acta* 21: 234–244

Walker BJ, Ort DR. 2015. Improved method for measuring the apparent CO<sub>2</sub> photocompensation point resolves the impact of multiple internal conductances to CO<sub>2</sub> to net gas exchange. *Plant Cell Environ* 38:2462- 2474

Yin X, Struik PC, Romero P, Harbinson J, Evers JB, van der Putten PEL, Vos J. 2009. Using combined measurements of gas exchange and chlorophyll fluorescence to estimate parameters of a biochemical C<sub>3</sub> photosynthesis model: a critical appraisal and a new integrated approach applied to leaves in a wheat (*Triticum aestivum*) canopy. *Plant Cell Environ* 32:448-464

Yin X, Sun Z, Struik PC, Gu J. 2011. Evaluating a new method to estimate the rate of leaf respiration in the light by analysis of combined gas exchange and chlorophyll fluorescence measurements. *Journal of Experimental Botany* 62: 3489–3499

## Examples

```
# Walker & Ort (2015) model

library(broom)
library(dplyr)
library(photosynthesis)

acq_data = system.file("extdata", "A_Ci_Q_data_1.csv", package = "photosynthesis") |>
  read.csv()

fit = fit_photosynthesis(
  .data = acq_data,
  .photo_fun = "r_light",
  .model = "walker_ort_2015",
  .vars = list(.A = A, .Q = Qin, .C = Ci),
  C_upper = 300,
  # Irradiance levels used in experiment
  Q_levels = c(1500, 750, 375, 125, 100, 75, 50, 25),
)

# The 'fit' object inherits class 'lm' and many methods can be used

## Model summary:
summary(fit)

## Estimated parameters:
coef(fit)

## 95% confidence intervals:
## n.b. these confidence intervals are not correct because the regression is fit
## sequentially. It ignores the underlying data and uncertainty in estimates of
## slopes and intercepts with each A-C curve. Use '.method = "brms"' to properly
## calculate uncertainty.
confint(fit)
```

```
## Tidy summary table using 'broom::tidy()'
tidy(fit, conf.int = TRUE, conf.level = 0.95)

## Calculate residual sum-of-squares
sum(resid(fit)^2)

# Yin et al. (2011) model

fit = fit_photosynthesis(
  .data = acq_data,
  .photo_fun = "r_light",
  .model = "yin_etal_2011",
  .vars = list(.A = A, .phiPSII = PhiPS2, .Q = Qin),
  Q_lower = 20,
  Q_upper = 250
)

# The 'fit' object inherits class 'lm' and many methods can be used

## Model summary:
summary(fit)

## Estimated parameters:
coef(fit)

## 95% confidence intervals:
confint(fit)

## Tidy summary table using 'broom::tidy()'
tidy(fit, conf.int = TRUE, conf.level = 0.95)

## Calculate residual sum-of-squares
sum(resid(fit)^2)

# Kok (1956) model

fit = fit_photosynthesis(
  .data = acq_data,
  .photo_fun = "r_light",
  .model = "kok_1956",
  .vars = list(.A = A, .Q = Qin),
  Q_lower = 20,
  Q_upper = 150
)

# The 'fit' object inherits class 'lm' and many methods can be used

## Model summary:
summary(fit)

## Estimated parameters:
coef(fit)
```

```
## 95% confidence intervals:
confint(fit)

## Tidy summary table using 'broom::tidy()'
tidy(fit, conf.int = TRUE, conf.level = 0.95)

## Calculate residual sum-of-squares
sum(resid(fit)^2)
```

---

fit_r_light_kok	<i>Estimating light respiration</i>
-----------------	-------------------------------------

---

## Description

### [Deprecated]

Please use `fit_r_light2()`.

## Usage

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

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

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 R <sub>light</sub> (Kok & Yin)
PPFD_upper	Upper light intensity limit for estimating R <sub>light</sub> (Kok & Yin)
P	Atmospheric pressure in kPa (Walker & Ort, 2015)
C_i_threshold	Threshold C <sub>i</sub> (in umol / mol) 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 CO<sub>2</sub> assimilation at very low light intensities and extrapolating from data above the breakpoint to estimate light respiration as the y-intercept. r<sub>light</sub> value should be negative, denoting an efflux of CO<sub>2</sub>.

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 A<sub>Ci</sub> 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<sup>-1</sup>, and respiration is output in umol m<sup>-2</sup> s<sup>-1</sup> 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<sub>i</sub>, the output value is technically C<sub>istar</sub>. You need to use C<sub>c</sub> 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 CO<sub>2</sub>.

## References

- Kok B. 1956. On the inhibition of photosynthesis by intense light. *Biochimica et Biophysica Acta* 21: 234–244
- Walker BJ, Ort DR. 2015. Improved method for measuring the apparent CO<sub>2</sub> photocompensation point resolves the impact of multiple internal conductances to CO<sub>2</sub> to net gas exchange. *Plant Cell Environ* 38:2462- 2474
- Yin X, Struik PC, Romero P, Harbinson J, Evers JB, van der Putten PEL, Vos J. 2009. Using combined measurements of gas exchange and chlorophyll fluorescence to estimate parameters of a biochemical C<sub>3</sub> photosynthesis model: a critical appraisal and a new integrated approach applied to leaves in a wheat (*Triticum aestivum*) canopy. *Plant Cell Environ* 32:448-464
- Yin X, Sun Z, Struik PC, Gu J. 2011. Evaluating a new method to estimate the rate of leaf respiration in the light by analysis of combined gas exchange and chlorophyll fluorescence measurements. *Journal of Experimental Botany* 62: 3489–3499

## 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",
    PPFDF = "Qin"
  ),
  PPFDF_lower = 20,
  PPFDF_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",
    PPFDF = "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	<i>Fitting temperature responses</i>
----------------	--------------------------------------

---

### 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.
model	Which temperature response model do you want to use? Defaults to all: Arrhenius, Heskel, Kruse, Medlyn, MMRT, Quadratic, and Topt.
start	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.
setvar	Which variable to set as constant for the Medlyn model? Defaults to "none", while "Hd" and "dS" options are available.

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

- Arrhenius S. 1915. Quantitative laws in biological chemistry. Bell.
- Heskel MA, O'Sullivan OS, Reich PB, Tjoelker MG, Weerasinghe LK, Penillard A, Egerton JJG, Creek D, Bloomfield KJ, Xiang J, Sinca F, Stangl ZR, la Torre AM, Griffin KL, Huntingford C, Hurry V, Meir P, Turnbull MH, Atkin OK. 2016. Convergence in the temperature response of leaf respiration across biomes and plant functional types. PNAS 113:3832-3837
- Hobbs JK, Jiao W, Easter AD, Parker EJ, Schipper LA, Arcus VL. 2013. Change in heat capacity for enzyme catalysis determines temperature dependence of enzyme catalyzed rates. ACS Chemical Biology 8:2388-2393.
- Kruse J, Adams MA. 2008. Three parameters comprehensively describe the temperature response of respiratory oxygen reduction. Plant Cell Environ 31:954-967
- Liang LL, Arcus VL, Heskel MA, O'Sullivan OS, Weerasinghe LK, Creek D, Egerton JJG, Tjoelker MG, Atkin OK, Schipper LA. 2018. Macromolecular rate theory (MMRT) provides a thermodynamics rationale to underpin the convergent temperature response in plant leaf respiration. Glob Chang Biol 24:1538-1547
- Medlyn BE, Dreyer E, Ellsworth D, Forstreuter M, Harley PC, Kirschbaum MUF, Le Roux X, Montpied P, Strassmeyer J, Walcroft A, Wang K, Loutstau D. 2002. Temperature response of parameters of a biochemically based model of photosynthesis. II. A review of experimental data. Plant Cell Environ 25:1167-1179

### Examples

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

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:**

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

where:

$$K_m = K_C(1 + O/K_O)$$

**RuBP regeneration-limited assimilation rate:**

$$W_{\text{regen}} = JC_{\text{chl}}/(4C_{\text{chl}} + 8\Gamma^*)$$

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

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

**TPU-limited assimilation rate:**

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

<i>Symbol</i>	<i>R</i>	<i>Description</i>	<i>Units</i>	<i>Default</i>
$C_{\text{chl}}$	C_chl	chloroplasic CO2 concentration	Pa	input
$\Gamma^*$	gamma_star	chloroplasic CO2 compensation point (T_leaf)	Pa	calculated
$J_{\text{max}}$	J_max	potential electron transport (T_leaf)	$\mu\text{mol CO}_2 / (\text{m}^2 \text{ s})$	calculated
$K_C$	K_C	Michaelis constant for carboxylation (T_leaf)	$\mu\text{mol} / \text{mol}$	calculated
$K_O$	K_O	Michaelis constant for oxygenation (T_leaf)	$\mu\text{mol} / \text{mol}$	calculated
$O$	0	atmospheric O2 concentration	kPa	21.27565
$\phi_J$	phi_J	initial slope of the response of J to PPFD	none	0.331
PPFD	PPFD	photosynthetic photon flux density	$\mu\text{mol quanta} / (\text{m}^2 \text{ s})$	1500
$R_d$	R_d	nonphotorespiratory CO2 release (T_leaf)	$\mu\text{mol CO}_2 / (\text{m}^2 \text{ s})$	calculated
$\theta_J$	theta_J	curvature factor for light-response curve	none	0.825
$V_{c,\text{max}}$	V_cmax	maximum rate of carboxylation (T_leaf)	$\mu\text{mol CO}_2 / (\text{m}^2 \text{ s})$	calculated
$V_{\text{tpu}}$	V_tpu	rate of triose phosphate utilization (T_leaf)	$\mu\text{mol CO}_2 / (\text{m}^2 \text{ s})$	calculated

**Value**

A list of four values with units  $\mu\text{mol CO}_2 / (\text{m}^2 \text{ s})$  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

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

Farquhar GD, Caemmerer S, Berry JA. 1980. A biochemical model of photosynthetic CO<sub>2</sub> assimilation in leaves of C<sub>3</sub> species. *Planta* 149: 78–90.

## Examples

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

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

---

get_default_model	<i>Get default model</i>
-------------------	--------------------------

---

## Description

### [Experimental]

Get the name of the default model used for different plant ecophysiological data analysis methods implemented in **photosynthesis**. Currently only used for [fit\\_aq\\_response2\(\)](#) and [fit\\_r\\_light2\(\)](#).

### Light response models:

- `marshall_biscoe_1980()`: Non-rectangular hyperbolic model of light responses
- `photoinhibition()`: Non-rectangular hyperbolic model of light responses with photoinhibition of `k_sat` at increasing `Q_abs`

## Usage

```
get_default_model(.photo_fun)

get_all_models(method)

marshall_biscoe_1980(Q_abs, k_sat, phi_J, theta_J)

photoinhibition(Q_abs, k_sat, phi_J, theta_J, b_inh)
```

**Arguments**

.photo_fun	A character string of <b>photosynthesis</b> function to call. One of: aq_response, r_light.
method	A character string of the statistical method to use: 'ls' for least-squares and 'brms' for Bayesian model using <code>brms::brm()</code> . Default is 'ls'.
Q_abs	Absorbed light intensity ( $\mu\text{mol m}^{-2} \text{s}^{-1}$ )
k_sat	Light saturated rate of process k
phi_J	Quantum efficiency of process k
theta_J	Curvature of the light response
b_inh	Inhibition parameter

**Value**

A character string with name of model.

**Examples**

```
get_default_model("aq_response")
get_default_model("r_light")
```

---

gs\_mod\_ballberry      *Stomatal conductance models*

---

**Description**

Stomatal conductance models

**Usage**

```
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**

A_net	Net CO <sub>2</sub> assimilation in $\mu\text{mol m}^{-2} \text{s}^{-1}$
C_air	CO <sub>2</sub> concentration at the leaf surface in $\mu\text{mol mol}^{-1}$
RH	Relative humidity as a proportion
D0	Vapor pressure sensitivity of stomata (Leuning 1995)
VPD	Vapor pressure deficit (kPa)
g0	Optimization model intercept term (Medlyn et al. 2011)
g1	Optimization model slope term (Medlyn et al. 2011)
gk	Optimization model root term (Medlyn et al. 2011)

**Value**

gs\_mod\_ballberry is used for fitting the Ball et al. (1987) model of stomatal conductance

gs\_mod\_leuning is used for fitting the Leuning (1995) model of stomatal conductance

gs\_mod\_opti fits the optimal stomatal conductance model according to Medlyn et al. 2011

gs\_mod\_optifull fits the full optimal stomatal conductance model according to Medlyn et al. 2011

**References**

Ball JT, Woodrow IE, Berry JA. 1987. A model predicting stomatal conductance and its contribution to the control of photosynthesis under different environmental conditions, in Progress in Photosynthesis Research, Proceedings of the VII International Congress on Photosynthesis, vol. 4, edited by I. Biggins, pp. 221–224, Martinus Nijhoff, Dordrecht, Netherlands.

Leuning R. 1995. A critical appraisal of a coupled stomatal- photosynthesis model for C3 plants. 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

---

 J

*J: Rate of electron transport (umol/m<sup>2</sup>/s)*

---

**Description**

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

**Usage**

J(pars, unitless = FALSE)

**Arguments**

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

**Details**

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

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

Symbol	R	Description	Units	Default
$J_{\max}$	J_max	potential electron transport (T_leaf)	$\mu\text{mol CO}_2 / (\text{m}^2 \text{ s})$	calculated
$\phi_J$	phi_J	initial slope of the response of J to PPFD	none	0.331

PPFD	PPFD	photosynthetic photon flux density	$\mu\text{mol quanta} / (\text{m}^2 \text{ s})$	1500
$\theta_J$	theta_J	curvature factor for light-response curve	none	0.825

**Value**

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

**Examples**

```
library(magrittr)
library(photosynthesis)

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

pars = c(leaf_par, enviro_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.

---

make\_parameters      *Make lists of parameters for photosynthesis*

---

## Description

Make lists of parameters for photosynthesis

make\_leafpar

make\_enviropar

make\_bakepar

make\_constants

## Usage

```
make_leafpar(replace = NULL, use_tealeaves)
```

```
make_enviropar(replace = NULL, use_tealeaves)
```

```
make_bakepar(replace = NULL)
```

```
make_constants(replace = NULL, use_tealeaves)
```

## Arguments

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

use\_tealeaves    Logical. Should leaf energy balance be used to calculate leaf temperature (T\_leaf)? If TRUE, `tleaf()` calculates T\_leaf. If FALSE, user-defined T\_leaf is used. Additional parameters and constants are required, see [make\\_parameters\(\)](#).

## Details

### Constants:

Symbol	R	Description	Units	Default
$D_{c,0}$	D_c0	diffusion coefficient for CO <sub>2</sub> in air at 0 °C	m <sup>2</sup> / s	$1.29 \times 10^{-5}$
$D_{h,0}$	D_h0	diffusion coefficient for heat in air at 0 °C	m <sup>2</sup> / s	$1.90 \times 10^{-5}$
$D_{m,0}$	D_m0	diffusion coefficient for momentum in air at 0 °C	m <sup>2</sup> / s	$1.33 \times 10^{-5}$
$D_{w,0}$	D_w0	diffusion coefficient for water vapor in air at 0 °C	m <sup>2</sup> / s	$2.12 \times 10^{-5}$
$\epsilon$	epsilon	ratio of water to air molar masses	none	0.622
$G$	G	gravitational acceleration	m / s <sup>2</sup>	9.8
$eT$	eT	exponent for temperature dependence of diffusion	none	1.75
$R$	R	ideal gas constant	J / mol / K	8.31
$\sigma$	sigma	Stephan-Boltzmann constant	W / m <sup>2</sup> / K <sup>4</sup>	$5.67 \times 10^{-8}$
$f_{Sh}$	f_sh	function to calculate constant(s) for Sherwood number	none	NA
$f_{Nu}$	f_nu	function to calculate constant(s) for Nusselt number	none	NA

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

Symbol	R	Description	Units	Default
$D_{s,gmc}$	Ds_gmc	empirical temperature response parameter	J / mol / K	487
$D_{s,Jmax}$	Ds_Jmax	empirical temperature response parameter	J / mol / K	388
$E_{a,\Gamma^*}$	Ea_gammapstar	empirical temperature response parameter	J / mol	24500
$E_{a,gmc}$	Ea_gmc	empirical temperature response parameter	J / mol	68900
$E_{a,Jmax}$	Ea_Jmax	empirical temperature response parameter	J / mol	56100
$E_{a,KC}$	Ea_KC	empirical temperature response parameter	J / mol	81000
$E_{a,KO}$	Ea_KO	empirical temperature response parameter	J / mol	23700
$E_{a,Rd}$	Ea_Rd	empirical temperature response parameter	J / mol	40400
$E_{a,Vcmax}$	Ea_Vcmax	empirical temperature response parameter	J / mol	52200
$E_{a,Vtpu}$	Ea_Vtpu	empirical temperature response parameter	J / mol	52200
$E_{d,gmc}$	Ed_gmc	empirical temperature response parameter	J / mol	149000
$E_{d,Jmax}$	Ed_Jmax	empirical temperature response parameter	J / mol	121000

**Environment parameters:**

Symbol	R	Description	Units	Default
$C_{air}$	C_air	atmospheric CO2 concentration	umol/mol	420
$O$	O	atmospheric O2 concentration	mol/mol	0.21
$P$	P	atmospheric pressure	kPa	101
PPFD	PPFD	photosynthetic photon flux density	umol / m <sup>2</sup> / s	1500
RH	RH	relative humidity	none	0.5
$u$	wind	windspeed	m / s	2

**Leaf parameters:**

Symbol	R	Description	Units	Default
$d$	leafsize	leaf characteristic dimension	m	0.1
$\Gamma^*$	gamma_star	chloroplastic CO2 compensation point (T_leaf)	umol/mol	NA
$\Gamma^*_{25}$	gamma_star25	chloroplastic CO2 compensation point (25 °C)	umol/mol	37.9
$g_{mc}$	g_mc	mesophyll conductance to CO2 (T_leaf)	mol / m <sup>2</sup> / s	NA
$g_{mc,25}$	g_mc25	mesophyll conductance to CO2 (25 °C)	mol / m <sup>2</sup> / s	0.4
$g_{sc}$	g_sc	stomatal conductance to CO2	mol / m <sup>2</sup> / s	0.4
$g_{uc}$	g_uc	cuticular conductance to CO2	mol / m <sup>2</sup> / s	0.01
$J_{max,25}$	J_max25	potential electron transport (25 °C)	umol / m <sup>2</sup> / s	200
$J_{max}$	J_max	potential electron transport (T_leaf)	umol / m <sup>2</sup> / s	NA
$k_{mc}$	k_mc	partition of g_mc to lower mesophyll	none	1
$k_{sc}$	k_sc	partition of g_sc to lower surface	none	1
$k_{uc}$	k_uc	partition of g_uc to lower surface	none	1
$K_{C,25}$	K_C25	Michaelis constant for carboxylation (25 °C)	umol / mol	268
$K_C$	K_C	Michaelis constant for carboxylation (T_leaf)	umol / mol	NA
$K_{O,25}$	K_O25	Michaelis constant for oxygenation (25 °C)	umol / mol	165000
$K_O$	K_O	Michaelis constant for oxygenation (T_leaf)	umol / mol	NA
$\phi_J$	phi_J	initial slope of the response of J to PPFD	none	0.331

$R_{d,25}$	R_d25	nonphotorespiratory CO2 release (25 °C)	umol / m <sup>2</sup> / s	2
$R_d$	R_d	nonphotorespiratory CO2 release (T_leaf)	umol / m <sup>2</sup> / s	NA
$\theta_J$	theta_J	curvature factor for light-response curve	none	0.825
$T_{leaf}$	T_leaf	leaf temperature	K	298
$V_{c,max,25}$	V_cmax25	maximum rate of carboxylation (25 °C)	umol / m <sup>2</sup> / s	150
$V_{c,max}$	V_cmax	maximum rate of carboxylation (T_leaf)	umol / m <sup>2</sup> / s	NA
$V_{tpu,25}$	V_tpu25	rate of triose phosphate utilization (25 °C)	umol / m <sup>2</sup> / s	200
$V_{tpu}$	V_tpu	rate of triose phosphate utilisation (T_leaf)	umol / m <sup>2</sup> / s	NA

If use\_tealeaves = TRUE, additional parameters are:

#### Constants:

Symbol	R	Description	Units	Default
$c_p$	c_p	heat capacity of air	J / g / K	1.01
$R_{air}$	R_air	specific gas constant for dry air	J / kg / K	287

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

Symbol	R	Description	Units	Default
--------	---	-------------	-------	---------

#### Environment parameters:

Symbol	R	Description	Units	Default
$E_q$	E_q	energy per mole quanta	kJ / mol	220
$f_{PAR}$	f_par	fraction of incoming shortwave radiation that is photosynthetically active radiation (PAR)	none	0.5
$r$	r	reflectance for shortwave irradiance (albedo)	none	0.2
$T_{air}$	T_air	air temperature	K	298
$T_{sky}$	T_sky	sky temperature	K	NA

#### Leaf parameters:

Symbol	R	Description	Units	Default
$\alpha_l$	abs_l	absorbivity of longwave radiation (4 - 80 um)	none	0.97
$\alpha_s$	abs_s	absorbivity of shortwave radiation (0.3 - 4 um)	none	0.5
$g_{sw}$	g_sw	stomatal conductance to H2O	mol / m <sup>2</sup> / s	NA
$g_{uw}$	g_uw	cuticular conductance to H2O	mol / m <sup>2</sup> / s	NA
$\text{logit}(sr)$	logit_sr	stomatal ratio (logit transformed)	none	NA

#### Optional leaf parameters:

Symbol	R	Description	Units	Default
--------	---	-------------	-------	---------

$\delta_{ias,lower}$	delta_ias_lower	effective distance through lower internal airspace	um	NA
$\delta_{ias,upper}$	delta_ias_upper	effective distance through upper internal airspace	um	NA
$A_{mes}/A$	A_mes_A	mesophyll surface area per unit leaf area	none	NA
$g_{liq,c,25}$	g_liqc25	liquid-phase conductance to CO2 (25 °C)	mol / m <sup>2</sup> / s	NA
$g_{liq,c}$	g_liqc	liquid-phase conductance to CO2 (T_leaf)	mol / m <sup>2</sup> / s	NA
$g_{ias,c,lower}$	g_iasc_lower	internal airspace conductance to CO2 in lower part of leaf (T_leaf)	mol / m <sup>2</sup> / s	NA
$g_{ias,c,upper}$	g_iasc_upper	internal airspace conductance to CO2 in upper part of leaf (T_leaf)	mol / m <sup>2</sup> / s	NA

## Value

make\_leafpar: An object inheriting from class `leaf_par()`  
 make\_enviropar: An object inheriting from class `enviro_par()`  
 make\_bakepar: An object inheriting from class `bake_par()`  
 make\_constants: An object inheriting from class `constants()`

## References

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

## Examples

```

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

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

```

---

parameter\_names      *Get vector of parameter names*

---

## Description

Get vector of parameter names

## Usage

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

**Arguments**

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

**Value**

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

**Examples**

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

---

photosynthesis	<i>Simulate C3 photosynthesis</i>
----------------	-----------------------------------

---

**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,
  use_tealeaves,
```

```

    quiet = FALSE,
    assert_units = TRUE,
    check = TRUE,
    prepare_for_tleaf = use_tealeaves,
    use_legacy_version = FALSE
)

```

### Arguments

<code>leaf_par</code>	A list of leaf parameters inheriting class <code>leaf_par</code> . This can be generated using the <code>make_leafpar</code> function.
<code>enviro_par</code>	A list of environmental parameters inheriting class <code>enviro_par</code> . This can be generated using the <code>make_enviropar</code> function.
<code>bake_par</code>	A list of temperature response parameters inheriting class <code>bake_par</code> . This can be generated using the <code>make_bakepar</code> function.
<code>constants</code>	A list of physical constants inheriting class <code>constants</code> . This can be generated using the <code>make_constants</code> function.
<code>use_tealeaves</code>	Logical. Should leaf energy balance be used to calculate leaf temperature ( <code>T_leaf</code> )? If TRUE, <code>tleaf()</code> calculates <code>T_leaf</code> . If FALSE, user-defined <code>T_leaf</code> is used. Additional parameters and constants are required, see <code>make_parameters()</code> .
<code>progress</code>	Logical. Should a progress bar be displayed?
<code>quiet</code>	Logical. Should messages be displayed?
<code>assert_units</code>	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.
<code>check</code>	Logical. Should arguments checks be done? This is intended to be disabled when <code>photo()</code> is called from <code>photosynthesis()</code> Default is TRUE.
<code>parallel</code>	Logical. Should parallel processing be used via <code>furrr::future_map()</code> ?
<code>use_legacy_version</code>	Logical. Should legacy model (<2.1.0) be used? See <b>NEWS</b> for further information. Default is FALSE.
<code>prepare_for_tleaf</code>	Logical. Should arguments additional calculations for <code>tleaf()</code> ? This is intended to be disabled when <code>photo()</code> is called from <code>photosynthesis()</code> . Default is <code>use_tealeaves</code> .

### 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:**

Symbol	R	Description	Units	Default
$D_{c,0}$	D_c0	diffusion coefficient for CO2 in air at 0 °C	m <sup>2</sup> / s	$1.29 \times 10^{-5}$
$D_{h,0}$	D_h0	diffusion coefficient for heat in air at 0 °C	m <sup>2</sup> / s	$1.90 \times 10^{-5}$
$D_{m,0}$	D_m0	diffusion coefficient for momentum in air at 0 °C	m <sup>2</sup> / s	$1.33 \times 10^{-5}$
$D_{w,0}$	D_w0	diffusion coefficient for water vapor in air at 0 °C	m <sup>2</sup> / s	$2.12 \times 10^{-5}$
$\epsilon$	epsilon	ratio of water to air molar masses	none	0.622
$G$	G	gravitational acceleration	m / s <sup>2</sup>	9.8
$eT$	eT	exponent for temperature dependence of diffusion	none	1.75
$R$	R	ideal gas constant	J / mol / K	8.31
$\sigma$	sigma	Stephan-Boltzmann constant	W / m <sup>2</sup> / K <sup>4</sup>	$5.67 \times 10^{-8}$
$f_{Sh}$	f_sh	function to calculate constant(s) for Sherwood number	none	NA
$f_{Nu}$	f_nu	function to calculate constant(s) for Nusselt number	none	NA
$D_{s,gmc}$	Ds_gmc	empirical temperature response parameter	J / mol / K	487
$D_{s,Jmax}$	Ds_Jmax	empirical temperature response parameter	J / mol / K	388
$E_{a,\Gamma^*}$	Ea_gammastar	empirical temperature response parameter	J / mol	24500
$E_{a,gmc}$	Ea_gmc	empirical temperature response parameter	J / mol	68900
$E_{a,Jmax}$	Ea_Jmax	empirical temperature response parameter	J / mol	56100
$E_{a,KC}$	Ea_KC	empirical temperature response parameter	J / mol	81000
$E_{a,KO}$	Ea_KO	empirical temperature response parameter	J / mol	23700
$E_{a,Rd}$	Ea_Rd	empirical temperature response parameter	J / mol	40400
$E_{a,Vcmax}$	Ea_Vcmax	empirical temperature response parameter	J / mol	52200
$E_{a,Vtpu}$	Ea_Vtpu	empirical temperature response parameter	J / mol	52200
$E_{d,gmc}$	Ed_gmc	empirical temperature response parameter	J / mol	149000
$E_{d,Jmax}$	Ed_Jmax	empirical temperature response parameter	J / mol	121000
$C_{air}$	C_air	atmospheric CO2 concentration	umol/mol	420
$O$	O	atmospheric O2 concentration	mol/mol	0.21
$P$	P	atmospheric pressure	kPa	101
PPFD	PPFD	photosynthetic photon flux density	umol / m <sup>2</sup> / s	1500
RH	RH	relative humidity	none	0.5
$u$	wind	windspeed	m / s	2
$d$	leafsize	leaf characteristic dimension	m	0.1
$\Gamma^*_{25}$	gamma_star25	chloroplastic CO2 compensation point (25 °C)	umol/mol	37.9
$g_{mc,25}$	g_mc25	mesophyll conductance to CO2 (25 °C)	mol / m <sup>2</sup> / s	0.4
$g_{sc}$	g_sc	stomatal conductance to CO2	mol / m <sup>2</sup> / s	0.4
$g_{uc}$	g_uc	cuticular conductance to CO2	mol / m <sup>2</sup> / s	0.01
$J_{max,25}$	J_max25	potential electron transport (25 °C)	umol / m <sup>2</sup> / s	200
$k_{mc}$	k_mc	partition of g_mc to lower mesophyll	none	1
$k_{sc}$	k_sc	partition of g_sc to lower surface	none	1
$k_{uc}$	k_uc	partition of g_uc to lower surface	none	1
$K_{C,25}$	K_C25	Michaelis constant for carboxylation (25 °C)	umol / mol	268
$K_{O,25}$	K_O25	Michaelis constant for oxygenation (25 °C)	umol / mol	165000
$\phi_J$	phi_J	initial slope of the response of J to PPFD	none	0.331
$R_{d,25}$	R_d25	nonphotorespiratory CO2 release (25 °C)	umol / m <sup>2</sup> / s	2

$\theta_J$	theta_J	curvature factor for light-response curve	none	0.825
$T_{\text{leaf}}$	T_leaf	leaf temperature	K	298
$V_{C,\text{max},25}$	V_cmax25	maximum rate of carboxylation (25 °C)	$\mu\text{mol} / \text{m}^2 / \text{s}$	150
$V_{\text{tpu},25}$	V_tpu25	rate of triose phosphate utilization (25 °C)	$\mu\text{mol} / \text{m}^2 / \text{s}$	200
$\delta_{\text{ias,lower}}$	delta_ias_lower	effective distance through lower internal airspace	$\mu\text{m}$	NA
$\delta_{\text{ias,upper}}$	delta_ias_upper	effective distance through upper internal airspace	$\mu\text{m}$	NA
$A_{\text{mes}}/A$	A_mes_A	mesophyll surface area per unit leaf area	none	NA
$g_{\text{liq,c},25}$	g_liqc25	liquid-phase conductance to CO2 (25 °C)	$\text{mol} / \text{m}^2 / \text{s}$	NA

**Baked Inputs:**

Symbol	R	Description	Units	Default
$\Gamma^*$	gamma_star	chloroplastic CO2 compensation point (T_leaf)	$\mu\text{mol}/\text{mol}$	NA
$g_{\text{mc}}$	g_mc	mesophyll conductance to CO2 (T_leaf)	$\text{mol} / \text{m}^2 / \text{s}$	NA
$J_{\text{max}}$	J_max	potential electron transport (T_leaf)	$\mu\text{mol} / \text{m}^2 / \text{s}$	NA
$K_C$	K_C	Michaelis constant for carboxylation (T_leaf)	$\mu\text{mol} / \text{mol}$	NA
$K_O$	K_O	Michaelis constant for oxygenation (T_leaf)	$\mu\text{mol} / \text{mol}$	NA
$R_d$	R_d	nonphotorespiratory CO2 release (T_leaf)	$\mu\text{mol} / \text{m}^2 / \text{s}$	NA
$V_{C,\text{max}}$	V_cmax	maximum rate of carboxylation (T_leaf)	$\mu\text{mol} / \text{m}^2 / \text{s}$	NA
$V_{\text{tpu}}$	V_tpu	rate of triose phosphate utilisation (T_leaf)	$\mu\text{mol} / \text{m}^2 / \text{s}$	NA
$g_{\text{liq,c}}$	g_liqc	liquid-phase conductance to CO2 (T_leaf)	$\text{mol} / \text{m}^2 / \text{s}$	NA
$g_{\text{ias,c,lower}}$	g_iasc_lower	internal airspace conductance to CO2 in lower part of leaf (T_leaf)	$\text{mol} / \text{m}^2 / \text{s}$	NA
$g_{\text{ias,c,upper}}$	g_iasc_upper	internal airspace conductance to CO2 in upper part of leaf (T_leaf)	$\text{mol} / \text{m}^2 / \text{s}$	NA

**Output:**

A	photosynthetic rate at C_ch1 ( $\mu\text{mol} \text{CO}_2 / \text{m}^2 / \text{s}$ )
C_ch1	chloroplastic CO2 concentration where A_supply intersects A_demand ( $\mu\text{mol} / \text{mol}$ )
C_i	intercellular CO2 concentration where A_supply intersects A_demand ( $\mu\text{mol} / \text{mol}$ )
g_tc	total conductance to CO2 at T_leaf ( $\text{mol} / \text{m}^2 / \text{s}$ )
value	A_supply - A_demand ( $\mu\text{mol} / (\text{m}^2 \text{s})$ ) at C_ch1
convergence	convergence code (0 = converged)

**Examples**

```
# Single parameter set with 'photo'

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

# Multiple parameter sets with 'photosynthesis'
```

```

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

```

---

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 73 rows and 11 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**

```
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**

```
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
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

*Read a LI-COR file*

---

## Description

### [Deprecated]

We are no longer updating this function. Please use [read\\_licor](#) instead.

## Usage

```
read_li6800(x)
```

**Arguments**

x                    File name

**Value**

Returns a data.frame from raw LI-COR files. Current support for LI-COR LI-6800 files only.

---

read_licor	<i>Read a LI-COR file</i>
------------	---------------------------

---

**Description****[Experimental]**

Reads a raw LI-COR LI6800 file, including remarks. This function was developed using output from Bluestem v.2.0.04 to v.2.1.08. We cannot guarantee backward compatibility with earlier versions of Bluestem. We will try to update code when new versions are released, but there maybe a time-lag, so inspect results carefully.

**Usage**

```
read_licor(
  file,
  bluestem_version = get_bluestem_version(file, n_max = 10L),
  ...
)
```

**Arguments**

file                    Path to a raw LI6800 file

bluestem\_version        Character string of Bluestem software version number. By default, the function will try to pull the version number from file.

...                     Argument passed to [read\\_lines](#)

**Value**

Returns a [tibble](#) from raw LI-COR LI6800 files.

---

required\_variables      *Variables required for **photosynthesis** models*

---

### Description

Variables required for **photosynthesis** models

### Usage

```
required_variables(.model, quiet)
```

### Arguments

.model                  A character string of model name to use. See [get\\_all\\_models\(\)](#).  
 quiet                  Flag. Should messages be suppressed? Default is FALSE.

---

simulate\_error          *Simulate gas exchange data with measurement error*

---

### Description

**[Experimental]**

### Usage

```
simulate_error(  
  ph_out,  
  chamber_pars,  
  n = 1L,  
  use_tealeaves = ("T_air" %in% colnames(ph_out))  
)
```

### Arguments

ph\_out                  A data frame of output from `photo()` or `photosynthesis()` with units.  
 chamber\_pars          A data frame with a single row of chamber parameters. See Note below for table of required parameters.  
 n                      Integer. Number of replicated simulations per row of `ph_out`.  
 use\_tealeaves          Flag. The **tealeaves** package uses a slightly different equation to calculate the saturating water content of air as a function temperature and pressure than LI-COR. If FALSE, the function uses LI-COR's equation in the LI6800 manual. If TRUE, it uses the **tealeaves** function for internal consistency. The function attempts to guess whether `ph_out` was run with **tealeaves**, but this can be manually overridden by providing a value for the argument.

**Value**

A data frame with  $n * \text{nrow}(\text{ph\_out})$  rows. It contains all the original output in `ph_out` as well as a column `.rep` indicating replicate number from 1 to  $n$ . Other new columns are assumed or measured chamber parameters and 'measured' values estimated from synthetic data with measurement error:

column name	assumed or derived?	description
<code>flow</code>	assumed	chamber flow rate
<code>leaf_area</code>	assumed	leaf area in chamber
<code>sigma_CO2_r</code>	assumed	standard deviation of measurement error in <code>CO2_r</code>
<code>sigma_CO2_s</code>	assumed	standard deviation of measurement error in <code>CO2_s</code>
<code>sigma_H2O_r</code>	assumed	standard deviation of measurement error in <code>H2O_r</code>
<code>sigma_H2O_s</code>	assumed	standard deviation of measurement error in <code>H2O_s</code>
<code>c_0</code>	derived	CO <sub>2</sub> concentration before entering chamber [ $\mu\text{mol} / \text{mol}$ ]
<code>w_i</code>	derived	Water vapor concentration within leaf [ $\text{mmol} / \text{mol}$ ]
<code>w_a</code>	derived	Water vapor concentration in chamber [ $\text{mmol} / \text{mol}$ ]
<code>w_0</code>	derived	Water vapor concentration before entering chamber [ $\text{mmol} / \text{mol}$ ]
<code>g_tw</code>	derived	Leaf conductance to water vapor [ $\text{mol}/\text{m}^2/\text{s}$ ]
<code>E_area</code>	derived	Evaporation rate per area [ $\text{mmol}/\text{m}^2/\text{s}$ ]
<code>E</code>	derived	Total evaporation rate [ $\text{mmol}/\text{s}$ ]
<code>CO2_r</code>	derived	CO <sub>2</sub> concentration before entering chamber with measurement error [ $\mu\text{mol} / \text{mol}$ ]
<code>CO2_s</code>	derived	CO <sub>2</sub> concentration in chamber with measurement error [ $\mu\text{mol} / \text{mol}$ ]
<code>H2O_s</code>	derived	Water vapor concentration in chamber with measurement error [ $\text{mmol} / \text{mol}$ ]
<code>H2O_r</code>	derived	Water vapor concentration before entering chamber with measurement error [ $\text{mmol} / \text{mol}$ ]
<code>E_meas</code>	derived	Total evaporation rate (measured) [ $\text{mmol}/\text{s}$ ]
<code>E_area_meas</code>	derived	Evaporation rate per area (measured) [ $\text{mmol}/\text{m}^2/\text{s}$ ]
<code>g_tw_meas</code>	derived	Leaf conductance to water vapor (measured) [ $\text{mol}/\text{m}^2/\text{s}$ ]
<code>g_sc_meas</code>	derived	Stomatal conductance to CO <sub>2</sub> (measured) [ $\text{mol}/\text{m}^2/\text{s}$ ]
<code>g_tc_meas</code>	derived	Leaf conductance to CO <sub>2</sub> (measured) [ $\text{mol}/\text{m}^2/\text{s}$ ]
<code>A_meas</code>	derived	Net photosynthetic CO <sub>2</sub> assimilation (measured) [ $\mu\text{mol}/\text{m}^2/\text{s}$ ]
<code>C_i</code>	derived	Intercellular CO <sub>2</sub> concentration (measured) [ $\mu\text{mol}/\text{mol}$ ]

**Note**

The required parameters for the `chamber_pars` argument are:

- `flow` [ $\mu\text{mol} / \text{s}$ ]: chamber flow rate
- `leaf_area` [ $\text{cm}^2$ ]: leaf area in chamber
- `sigma_CO2_s` [ $\mu\text{mol} / \text{mol}$ ]: standard deviation of sample [CO<sub>2</sub>] measurement error
- `sigma_CO2_r` [ $\mu\text{mol} / \text{mol}$ ]: standard deviation of reference [CO<sub>2</sub>]
- `sigma_H2O_s` [ $\text{mmol} / \text{mol}$ ]: standard deviation of sample [H<sub>2</sub>O] measurement error
- `sigma_H2O_r` [ $\text{mmol} / \text{mol}$ ]: standard deviation of sample [H<sub>2</sub>O] measurement error

Units for `flow` and `leaf_area` should be provided; units are implied for `sigma`'s but not necessary to specify because `rnorm()` drop units.

To evaluate the accuracy and precision of parameter estimation methods, it may be useful to simulate data with realistic measurement error. This function takes output from `photo()` or

photosynthesis() models, adds measurement error in CO<sub>2</sub> and H<sub>2</sub>O concentrations, and calculates parameter estimates with synthetic data. Currently, the function assumes a simplified 1-dimensional CO<sub>2</sub> and H<sub>2</sub>O conductance model: zero cuticular conductance, infinite boundary layer conductance, and infinite airspace conductance. Other assumptions include:

- chamber flow rate, leaf area, leaf temperature, and air pressure are known without error
- measurement error is normally distributed mean 0 and standard deviation specified in chamber\_pars

This function was designed with the LI-COR LI6800 instrument in mind, but in principle applies to any open path gas exchange system.

[CO<sub>e</sub>VJRVEZDCD9abtGBofcIo0mcCo2tpuCB-5-]: R:CO<sub>e</sub>VJRVEZDCD9abtGBofcIo0mcCo2tpuCB-5-%5C

## Examples

```
library(photosynthesis)

# Use photosynthesis() to simulate 'real' values
# `replace = ...` sets parameters to meet assumptions of `simulate_error()`
lp = make_leafpar(replace = list(
  g_sc = set_units(0.1, mol/m^2/s),
  g_uc = set_units(0, mol/m^2/s),
  k_mc = set_units(0, 1),
  k_sc = set_units(0, 1),
  k_uc = set_units(0, 1)
),
  use_tealeaves = FALSE)

ep = make_enviropar(replace = list(
  wind = set_units(Inf, m/s)
), use_tealeaves = FALSE)
bp = make_bakepar()
cs = make_constants(use_tealeaves = FALSE)

chamber_pars = data.frame(
  flow = set_units(600, umol / s),
  leaf_area = set_units(6, cm ^ 2),
  sigma_CO2_s = 0.1,
  sigma_CO2_r = 0.1,
  sigma_H2O_s = 0.1,
  sigma_H2O_r = 0.1
)

ph = photosynthesis(lp, ep, bp, cs, use_tealeaves = FALSE, quiet = TRUE) |>
  simulate_error(chamber_pars, n = 1L)
```

---

t\_response\_arrhenius *Temperature response functions*


---

**Description**

Temperature response functions

**Usage**

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 <sup>-1</sup> (Medlyn et al. 2002)
dEa	Temperature-dependent change in Ea in K <sup>2</sup> (Kruse et al. 2008)
Ea_ref	Activation energy in J mol <sup>-1</sup> (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 <sup>-1</sup> (Medlyn et al. 2002)
dS	Entropy parameter in J mol <sup>-1</sup> (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 <sup>-1</sup> K <sup>-1</sup> (Hobbs et al. 2013)
dG	Change in Gibbs free energy of the reaction at 25 C in J mol <sup>-1</sup> (Hobbs et al. 2013)
dH	Change in enthalpy of the reaction at 25 C in J mol <sup>-1</sup> (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**

Arrhenius S. 1915. Quantitative laws in biological chemistry. Bell.

Heskel et al. 2016. Convergence in the temperature response of leaf respiration across biomes and plant functional types. PNAS 113:3832-3837

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

Kruse J, Adams MA. 2008. Three parameters comprehensively describe the temperature response of respiratory oxygen reduction. Plant Cell Environ 31:954-967

Medlyn BE, Dreyer E, Ellsworth D, Forstreuter M, Harley PC, Kirschbaum MUF, Le Roux X, Montpied P, Strassmeyer J, Walcroft A, Wang K, Loutstau D. 2002. Temperature response of parameters of a biochemically based model of photosynthesis. II. A review of experimental data. Plant Cell Environ 25:1167-1179

# Index

- \* **datasets**
  - photo\_parameters, 65
- .get\_gbc (CO2\_conductance), 11
- .get\_gbc(), 13, 14
- .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, 5
  
- bake, 7, 14
- bake(), 9, 14, 62
- bake\_par, 10
- bake\_par(), 8, 60
- baked-class, 9
- brms::brm(), 26, 37, 41, 54
- brms::brmsfit(), 26, 41
  
- calculate\_j (calculate\_jmax), 11
- calculate\_jmax, 11
- calculated, 7, 13, 52, 55
- calculated-parameters, 10
- CO2\_conductance, 11
- compile\_data, 15
- compute\_sensitivity, 16
- constants, 18
- constants(), 60, 62
- cross\_df(), 62
  
- enviro\_par, 19
- enviro\_par(), 60, 62
  
- fit\_aci\_response, 19
- fit\_aq\_response, 23
- fit\_aq\_response2, 25
- fit\_aq\_response2(), 37, 53
- fit\_g\_mc\_variableJ, 30
  
- fit\_gs\_model, 27
- fit\_hydra\_vuln\_curve, 33
- fit\_many, 35
- fit\_photosynthesis, 37
- fit\_photosynthesis(), 25, 40
- fit\_PV\_curve, 38
- fit\_r\_light2, 40
- fit\_r\_light2(), 37, 53
- fit\_r\_light\_kok, 44
- fit\_r\_light\_WalkerOrt  
    (fit\_r\_light\_kok), 44
- fit\_r\_light\_yin (fit\_r\_light\_kok), 44
- fit\_t\_response, 47
- furrr::future\_map(), 62
- FvCB, 51
- FvCB(), 62
  
- get\_all\_models (get\_default\_model), 53
- get\_all\_models(), 26, 37, 41, 70
- get\_default\_model, 53
- get\_f\_parameter  
    (calculated-parameters), 10
- gs\_mod\_ballberry, 54
- gs\_mod\_leuning (gs\_mod\_ballberry), 54
- gs\_mod\_opti (gs\_mod\_ballberry), 54
- gs\_mod\_optifull (gs\_mod\_ballberry), 54
  
- J, 55
  
- leaf\_par, 56
- leaf\_par(), 8, 9, 60, 62
- list(), 9
  
- make\_bakepar (make\_parameters), 57
- make\_constants (make\_parameters), 57
- make\_constants(), 8
- make\_enviropar (make\_parameters), 57
- make\_leafpar (make\_parameters), 57
- make\_parameters, 57
- make\_parameters(), 18, 19, 56, 57, 61, 62

map, [35](#)  
marshall\_biscoe\_1980  
    (get\_default\_model), [53](#)  
  
parameter\_names, [60](#)  
photo (photosynthesis), [61](#)  
photo(), [62](#)  
photo\_parameters, [65](#)  
photoinhibition (get\_default\_model), [53](#)  
photosynthesis, [10](#), [61](#)  
photosynthesis(), [62](#), [65](#)  
ppm2pa, [65](#)  
print\_graphs, [66](#)  
  
read\_li6800, [68](#)  
read\_licor, [68](#), [69](#)  
read\_lines, [69](#)  
required\_variables, [70](#)  
required\_variables(), [26](#), [37](#), [41](#)  
  
simulate\_error, [70](#)  
stats::lm(), [41](#)  
stats::nls(), [26](#), [41](#)  
  
t\_response\_arrhenius, [73](#)  
t\_response\_arrhenius\_kruse  
    (t\_response\_arrhenius), [73](#)  
t\_response\_arrhenius\_medlyn  
    (t\_response\_arrhenius), [73](#)  
t\_response\_arrhenius\_topt  
    (t\_response\_arrhenius), [73](#)  
t\_response\_calc\_dS  
    (t\_response\_arrhenius), [73](#)  
t\_response\_calc\_topt  
    (t\_response\_arrhenius), [73](#)  
t\_response\_heskel  
    (t\_response\_arrhenius), [73](#)  
t\_response\_mmrt (t\_response\_arrhenius),  
    [73](#)  
tealeaves::get\_Dx(), [14](#)  
temp\_resp1 (bake), [7](#)  
temp\_resp2 (bake), [7](#)  
tibble, [69](#)  
tleaf(), [18](#), [19](#), [56](#), [57](#), [61](#), [62](#)  
  
W\_carbox (FvCB), [51](#)  
W\_regen (FvCB), [51](#)  
W\_tpu (FvCB), [51](#)