Package 'rbcmodel'

August 4, 2026

Title Model Rubisco Carboxylation Rate Across Temperature, CO2, and O2

Version 1.0.1

Description Collects published kinetics of ribulose 1,5-bisphosphate carboxylase/oxygenase (Rubisco) and uses these kinetics to model the carboxylation rate across CO2 and O2 concentrations and different temperatures. The carboxylation rate can be modeled as the gross rate or the net rate, which takes into account the oxygenase activity of the enzyme using one of the three known phosphoglycolate salvage pathways. Custom enzymes and temperature dependences can be created with user kinetics, or published kinetics can be accessed through a meta-analysis contained within the package. An expansion of methods from Harrison et al. (2025) <doi:10.1128/aem.00604-25>.

License MIT + file LICENSE

Encoding UTF-8

Imports plot3D

Depends R (>= 3.5)

LazyData true

Suggests knitr, rmarkdown

VignetteBuilder knitr

Config/roxygen2/version 8.0.0

RoxygenNote 8.0.0

NeedsCompilation no

Author Kaitlin Harrison [aut, cre, cph],
Wing-Ho Ko [aut]

Maintainer Kaitlin Harrison <k.harrison.sci@gmail.com>

Repository CRAN

Date/Publication 2026-08-04 13:50:02 UTC

Contents

`check_DHScale` *Validator for the S3 class 'DHScale'*

Description

Throw error if any scaling constants are non-numeric, or if name is not a character string.

Usage

```
check_DHScale(the_DHScale)
```

Arguments

the_DHScale	the 'DHScale' instance to be checked.

Value

the supplied DHScale instance, if all checks pass.

Examples

```
bad_DHScale <- new_DHScale(NA, 41, 26, -13.8, name="bad") # bad instance
try(check_DHScale(bad_DHScale)) # produces error
```

---

check_enzyme *Validator for the S3 class 'enzyme'*

Description

Throw error if any kinetic values or temperature are non-numeric, or if name is not a character string.

Usage

```
check_enzyme(the_enzyme)
```

Arguments

the_enzyme	the 'enzyme' instance to be checked.

Value

the supplied enzyme instance, if all checks pass.

Examples

```
bad_enzyme <- new_enzyme(NA, 144, 394, 10.8, name="bad") # bad instance
try(check_enzyme(bad_enzyme)) # produces error
```

cite_Rbc *Cite data collected in the package.*

Description

Cite data collected in the package.

Usage

```
cite_Rbc(identifier, type = "full")
```

Arguments

identifier	The identifier of the data. Takes a single identifier or a list of identifiers.
type	The type of citation desired. Can take "full", which will return the full citation, or "short", which returns the pmid or doi, if available.

Value

For data from a single study, this returns the citation for that study. For data averaged or compiled by this study, returns a note to cite this package.

Examples

```
cite_Rbc("average_1B_all")
cite_Rbc(c("japonica_Sage_2002b","japonica_orr_2016_dH","average_1B_C3_warm_dH"))
cite_Rbc("breve_banda_2020",type="short")
```

CO2_comparison *Creates a combined model that compares two different enzymes*

Description

Creates a combined model that compares two different enzymes. Requires the input of two models output from the CO2_dependence equation.

Usage

```
CO2_comparison(model1, model2)
```

Arguments

model1	The first model to compare. Positive values from the output equation will mean that this modeled enzyme is faster.
model2	The second model to compare. Negative values from the output equation will mean that this modeled enzyme is faster.

Value

A function with the same inputs as the original models comparing the two models.

Examples

```
pro_enzyme <- new_enzyme(6.6, 144, 394, 10.8, name="pro")
bacIA_DHScale <- new_DHScale(47.2,40.8,26.7,-21.8,name="bacIA")
pro_model <- CO2_dependence(pro_enzyme, bacIA_DHScale, "canon") #first model
tmrIA_DHScale <- modify_DHScale(bacIA_DHScale,kcat_dH=32,name="tmr_IA")
tmrIA_model <- CO2_dependence(pro_enzyme, tmrIA_DHScale, "canon") #second model
tmr_vs_pro <- CO2_comparison(pro_model, tmrIA_model) #creating the comparison
print(tmr_vs_pro) # prints the comparison equation
# prints the amount for which the tmrIA_model is faster than the pro_model
tmr_vs_pro(126,54,15)
```

CO2_dependence *Function factory to model an enzyme's carbon output*

Description

Function factory to model an enzyme's carbon output versus temperature and CO2/O2 concentration. Uses a 'DHScale' object to scale each kinetic parameter from an 'enzyme' object in regards to temperature. Based on the stoichiometry of known phosphoglycolate salvage pathways, it also takes into account the carbon lost due to the Rubisco oxygenation reaction based on user input.

Usage

```
CO2_dependence(enzyme, DHScale, PGS = NULL)
```

Arguments

enzyme	An 'enzyme' object. (See enzyme class for more info.)
DHScale	A 'DHScale' object. (See DHScale class for more info.)
PGS	One of 'gross', 'canon', 'diatom', or 'alt'. This option allows you to pick a stoichiometry for the phosphoglycolate salvage (PGS) pathway. If NULL, the choice is inferred from the default PGS of the enzyme instance. If you do not want to incorporate PGS, use the 'gross' option, while 'canon' applies the C2 cycle/glycerate pathway stoichiometry (1 CO2 for every 2 2-phosphoglycolate created). 'alt' corresponds to the oxalyl-CoA or malate cycle pathways (2 CO2 for every 2-phosphoglycolate created). 'diatom' corresponds to the proposed diatom-specific pathways, which do not return 2-PG to the Calvin Cycle, but incorporate it directly into biomass with the exception of 1 CO2 for every 2-phosphoglycolate created.

Value

A new function that models carbon output with three variables: CO2 (in uM), O2 (in uM), and temperature (in celsius).

Examples

```
pro_enzyme <- new_enzyme(6.6, 144, 394, 10.8, name="pro")
bacIA_DHScale <- new_DHScale(47.2,40.8,26.7,-21.8,name="bacIA")
pro_model <- CO2_dependence(pro_enzyme, bacIA_DHScale, "canon") # sets up model
pro_model(126,54,15) # prints carbon fixation at 15 deg C, 126 uM CO2, 54 uM O2
```

DHScale *Create an DHScale instance by looking up the relevant data in database*

Description

Create an DHScale instance by looking up the relevant data in database

Usage

```
DHScale(id_name, id_col = "identifier", scale_name = NULL, data = NULL)
```

Arguments

id_name	the name of identifier used to pull out a single record out of the database
id_col	the name of the column from which the id_name will be searched for within the database
scale_name	the name of the DHScale instance. If NULL the value is defaulted to id_name
data	the database to search from. Can be "averaged", NULL, or a data.frame. If NULL or "averaged", the averaged table is searched. NOTE: if a custom data.frame is supplied, it is assumed to have columns named "kcat_dH", "Kc_dH", "Ko_dH", and "S_dH".

Value

a new DHScale instance

Examples

```
cyano <- DHScale("average_1Ac_Cyanobacteria_dH") # retrieved from averaged database
```

Enzyme *Create an Enzyme instance by looking up the relevant data in database*

Description

Create an Enzyme instance by looking up the relevant data in database

Usage

```
Enzyme(id_name, id_col = "identifier", enzyme_name = NULL, data = NULL)
```

Arguments

id_name	the name of identifier used to pull out a single record out of the database
id_col	the name of the column from which the id_name will be searched for within the database
enzyme_name	the name of the enzyme instance. If NULL the value is defaulted to id_name
data	the database to search from. Can be "abridged", "comprehensive", NULL, or a data.frame. If NULL, the abridged table is first searched and if no entry is found the comprehensive table is searched. NOTE: if a custom data.frame is supplied, it is assumed to have columns named "kcat_val", "Kc_val", "Ko_val", and "S_val". In addition, it may have optional columns "temp", "kcat_T", "Kc_T", "Ko_T", "S_T", and "PGS"

Value

a new Enzyme instance

Examples

```
biuncialis <- Enzyme("biuncialis_Prins_2016") # retrieved from abridged database
# specify enzyme using species name
vavilovii <- Enzyme("vavilovii", "species", data="comprehensive")
```

kinetics_T_dep *Function factory for calculating temperature-dependent kinetics*

Description

Calculates the non-supplied constant (c) for an Arrhenius-type temperature scaling function (as seen in Galmés et al. 2016). Uses measured values and the temperature they were measured at, along with a supplied temperature scaling constant (also called the activation energy, ΔH (in kJ/mol)).

Usage

```
kinetics_T_dep(std, delta_H, temp = 25)
```

Arguments

std	Measured value
delta_H	ΔH value for this kinetic parameter for this organism type (in kJ/mol).
temp	Temperature at which param 'std' was measured. Defaults to 25°C.

Value

A new function that scales the supplied kinetic parameter with temperature (in celsius).

Examples

```
kcat_proteo_Ia <- kinetics_T_dep(15, 44.7) # create new function
kcat_proteo_Ia(30) # evaluate the kinetic parameter at a different temperature
```

---

last_non_NA *Return the last non-NA value from a vector.*

Description

Return the last non-NA value from a vector.

Usage

```
last_non_NA(x)
```

Arguments

x	the vector to extract values from.

Value

The last non-NA value, or NA if none exists.

make_4D_grid *Construct a 4D grid*

Description

Construct a 4D grid, where the first 3 coordinates are independent variables, while the 4th coordinate is dependent variable arising from evaluating a function against the values of the first 3 coordinates.

Usage

```
make_4D_grid(func, x_seq, y_seq, z_seq, var_names = c("x", "y", "z", "f"))
```

Arguments

func	The function to apply on the grid points to obtain the value of the dependent variable.
x_seq	A vector of coordinates for the first independent variable.
y_seq	A vector of coordinates for the second independent variable.
z_seq	A vector of coordinates for the third independent variable.
var_names	Names for the 3 independent and 1 dependent variables, in a single character vector.

Details

The grid is represented by a 4-element list, where each member is a 3-dimensional array representing grid points. The first array is the value of first independent variable on each grid point, etc., and the fourth array is the value of function evaluated at each grid point.

Value

A 4D grid as a 4-element list of 3-dimensional arrays

Examples

```
f <- function(x, y, z) { x + y * z }
g1 <- make_4D_grid(f, 1:3, c(-2,2), seq(4, 10, 2))
g1
```

`mean_if_num` *Compute the mean of a vector if the vector is numerical or logical. Otherwise, return the common value if all entries take the same value, or NA in any other cases.*

Description

Compute the mean of a vector if the vector is numerical or logical. Otherwise, return the common value if all entries take the same value, or NA in any other cases.

Usage

```
mean_if_num(x, na.rm = TRUE)
```

Arguments

`x`	the vector to compute average from.
`na.rm`	whether NA's are ignored for the purpose of computing average or for checking consistency between entries.

Value

The average or common value, if exists.

`median_if_num` *Compute the median of a vector if the vector is numerical or logical. Otherwise, return the common value if all entries take the same value, or NA in any other cases.*

Description

Compute the median of a vector if the vector is numerical or logical. Otherwise, return the common value if all entries take the same value, or NA in any other cases.

Usage

```
median_if_num(x, na.rm = TRUE)
```

Arguments

`x`	the vector to compute average from.
`na.rm`	whether NA's are ignored for the purpose of computing average or for checking consistency between entries.

Value

The average or common value, if exists.

merge_entries *Merge (combine) several entries from a data.frame.*

Description

Merge (combine) several entries from a data.frame.

Usage

```
merge_entries(
  data,
  id_col,
  id_vals,
  method,
  new_id = NA,
  const_cols = NULL,
  keep_merged = TRUE,
  keep_unmerged = TRUE
)
```

Arguments

data	The data.frame from which entries are pulled.
id_col	The column from which the relevant entries are identified.
id_vals	The values from id_col for which the corresponding entries will be pulled.
method	The method to combine entries. Can be "average", "median", or "last".
new_id	The new identifier (for id_col) for the merged entry.
const_cols	The columns to check for consistency. Error is thrown if the values from the pulled entries are not identical for any such columns.
keep_merged	Whether to keep the original entries that are pulled and merged.
keep_unmerged	Whether to keep the original entries that have NOT been touched.

Value

a data frame consisting of the merged entries, and optionally the unmerged and/or the untouched entries.

Examples

```
df <- data.frame(
  id = 1:4,
  x = c(1, 2, NA, 3),
  y = c("this", "this", NA, "that")
)

df2 <- merge_entries(df, "id", c(1,4), "average", new_id = "new")
```

```
df2

df3 <- merge_entries(df, "id", c(4, 3, 1), "last", keep_merged = FALSE)
df3

try(merge_entries(df, "id", c(4, 3, 1), "last", const_cols="y")) # expects error
```

`modify_DHScale` *Create a new instance of S3 class 'DHScale', starting from a previous instance*

Description

If any argument (other than `the_DHScale`, which is required) is `NULL`, the corresponding instance attribute will copy its value from the previous instance. Otherwise, the input value will become the value of the corresponding attribute in the new instance. For detailed explanation of each 'DHScale' attribute, see the documentation of `new_DHScale()`

Usage

```
modify_DHScale(
  the_DHScale,
  kcat_dH = NULL,
  Kc_dH = NULL,
  Ko_dH = NULL,
  S_dH = NULL,
  name = NULL
)
```

Arguments

`the_DHScale`	the previous 'DHScale' instance to base the new instance on.
`kcat_dH`	if not NULL, the kcat_dH value for the new instance
`Kc_dH`	if not NULL, the Kc_dH value for the new instance
`Ko_dH`	if not NULL, the Ko_dH value for the new instance
`S_dH`	if not NULL, the S_dH value for the new instance
`name`	if not NULL, the identifier ("name") for the new instance

Value

a new instance of the S3 class 'DHScale'.

Examples

```
bacIA_DHScale <- new_DHScale(47.2,40.8,26.7,-21.8,name="bacIA") # new instance
print(bacIA_DHScale) # printout DHScale info
tmrIA_DHScale <- modify_DHScale(bacIA_DHScale,kcat_dH=32,name="tmr_IA") # modify instance
print(tmrIA_DHScale) # printout modified DHScale info
```

`modify_enzyme` *Create a new instance of S3 class 'enzyme' starting from a previous instance*

Description

If any argument (other than `the_enzyme`, which is required) is `NULL`, the corresponding instance attribute will copy its value from the previous instance. Otherwise, the input value will become the value of the corresponding attribute in the new instance. For detailed explanation of each 'enzyme' attribute, see the documentation of `new_enzyme()`

Usage

```
modify_enzyme(
  the_enzyme,
  kcat_val = NULL,
  Kc_val = NULL,
  Ko_val = NULL,
  S_val = NULL,
  temp = NULL,
  kcat_T = NULL,
  Kc_T = NULL,
  Ko_T = NULL,
  S_T = NULL,
  T_max = NULL,
  PGS = NULL,
  name = NULL
)
```

Arguments

`the_enzyme`	the previous 'enzyme' instance to base the new instance on.
`kcat_val`	if not NULL, the kcat_val value for the new instance
`Kc_val`	if not NULL, the Kc_val value for the new instance
`Ko_val`	if not NULL, the Ko_val value for the new instance
`S_val`	if not NULL, the S_val value for the new instance
`temp`	NOT USED. A warning is produced if this argument is used. Please set the temperature for each kinetic parameters separately
`kcat_T`	if not NULL, the kcat_T value for the new instance
`Kc_T`	if not NULL, the Kc_T value for the new instance
`Ko_T`	if not NULL, the Ko_T value for the new instance
`S_T`	if not NULL, the S_T value for the new instance
`T_max`	if not NULL, the upper thermal limit of the new instance
`PGS`	if not NULL, the default phosphoglycolate salvage (PGS) pathway for the new instance
`name`	if not NULL, the identifier ("name") for the new instance

Value

a new instance of the S3 class 'enzyme'.

Examples

```
pro_enzyme <- new_enzyme(6.6, 144, 394, 10.8, name="pro") # new instance
print(pro_enzyme) # printout enzyme info
pro_worse_enzyme <- modify_enzyme(pro_enzyme, kcat_val=2, Kc_val=310, name="pro_worse")
print(pro_worse_enzyme) # printout modified enzyme info
```

---

new_DHScale *Constructor for the S3 class 'DHScale'.*

Description

This class collects temperature scaling constants for each of the four kinetic parameters. These scaling constants are also called the activation energy of that kinetic parameter, and are typically denoted with ΔH. In combination with an enzyme-class object, a DHScale-class object can be used to create a function that models an enzyme's carbon output with regards to temperature and CO2/O2 concentration.

Usage

```
new_DHScale(kcat_dH, Kc_dH, Ko_dH, S_dH, name = "")
```

Arguments

kcat_dH	The temperature scaling constant of kcat for the carboxylation reaction, also known as the activation energy (in kJ/mol).
Kc_dH	The temperature scaling constant of Kc (in kJ/mol).
Ko_dH	The temperature scaling constant of Ko (in kJ/mol).
S_dH	The temperature scaling constant of Sc/o (in kJ/mol).
name	An identifier ("name") of the scale instance.

Value

A new 'DHScale' instance.

Examples

```
new_DHScale(40.1, 38.8, 26.7, -27.4, name = "Form Ia, cyanobacteria")
```

new_enzyme *Constructor for the S3 class 'enzyme'.*

Description

The class 'enzyme' is designed to encapsulate the information necessary to describe a particular measured enzyme's response to CO2, O2, and temperature. All of Rubisco's key kinetic parameters are used as inputs, along with the temperature at which they were measured. In combination with a DHScale-class object, an enzyme-class object can be used to create a function that models a Rubisco enzyme's carbon output with regards to temperature and CO2/O2 concentration.

Usage

```
new_enzyme(
  kcat_val,
  Kc_val,
  Ko_val,
  S_val,
  temp = 25,
  kcat_T = NULL,
  Kc_T = NULL,
  Ko_T = NULL,
  S_T = NULL,
  T_max = 55,
  PGS = NA,
  name = ""
)
```

Arguments

kcat_val	The measured value of kcat for the Rubisco carboxylase reaction (in s^-1). kcat represents the number of reactions per second a single active site can perform. The temperature at which it was measured can be found in kcat_T (if present) or temp.
Kc_val	The measured value of Kc (in uM). Kc is the Michaelis constant of Rubisco for CO2, or its affinity for CO2. The temperature at which it was measured can be found in Kc_T (if present) or temp.
Ko_val	The measured value of Ko (in uM). Ko is the Michaelis constant of Rubisco for O2, or its affinity for O2. The temperature at which it was measured can be found in Ko_T (if present) or temp.
S_val	The measured value of Sc/o (unitless). Sc/o is the specificity of Rubisco for CO2 versus O2. The temperature at which it was measured can be found in S_T (if present) or temp.
temp	The common temperature (in celsius) at which the kinetic constants are measured. Can be overridden by the more specialized arguments kcat_T, Kc_T, Ko_T, and S_T. If NULL these 4 arguments are used to specify temperatures.

kcat_T	The temperature (in celsius) at which kcat_val is measured.
Kc_T	The temperature (in celsius) at which Kc_val is measured.
Ko_T	The temperature (in celsius) at which Ko_val is measured.
S_T	The temperature (in celsius) at which S_val is measured.
T_max	The upper thermal limit (in celsius) of the enzyme.
PGS	The default phosphoglycolate salvage (PGS) pathway of the enzyme. Can be NA.
name	An identifier ("name") of the enzyme instance.

Value

A new enzyme instance.

Examples

```
new_enzyme(6.6, 144, 394, 10.8, name="pro")
```

---

permute_4D_grid *Permute the independent variables in a 4D grid created by make_4D_grid().*

Description

In the new grid, every element in the list is a permuted array. Moreover, the order of the independent variables in the list is also permuted.

Usage

```
permute_4D_grid(grid, perm)
```

Arguments

grid	The original grid to be permuted.
perm	the subscript permutation vector, a permutation of the integers c(1, 2, 3).

Value

A permuted 4D grid

Examples

```
f <- function(x, y, z) { x + y * z }
g1 <- make_4D_grid(f, 1:3, c(-2,2), seq(4, 10, 2))
g2 <- permute_4D_grid(g1, c(2,3,1))
g2
```

plot_slice_3D *Create a plot of the dependent variable in a 3D slice*

Description

Create a plot of the dependent variable (4-th index) in a 3D slice (created using, e.g., slice_4D_grid()) against its two non-constant independent dimensions. In the plot, the dependent variable is represented by contours and false color, while the two independent variables form the horizontal and the vertical axes.

Usage

```
plot_slice_3D(
  grid_slice,
  contours = NULL,
  cmin = NULL,
  cmax = NULL,
  dims = NULL,
  colors = "default",
  NA_color = "grey",
  contour_col = "black",
  xlabel = "",
  ylabel = "",
  clabel = c("Carbon", "(C/s)"),
  lwd = 2,
  ...
)
```

Arguments

grid_slice	The 3D slice to be plotted
contours	The values of dependent variable at which the contours are drawn, as a single numeric vector. If NULL no contours are produced.
cmin	The lower bound for the color mapping. If NULL defaults to the minimum value of the contour/color variable.
cmax	The upper bound for the color mapping. If NULL defaults to the maximum value of the contour/color variable.
dims	Length 3 integer vector, e.g., c(1,2,4), that specify the horizontal, vertical, and contour/color variables of the plot. If NULL, the function will attempt to automatically determine horizontal coordinate and the vertical coordinate, while the last element of the slice is treated as the dependent variable. Note that the horizontal coordinate should ALWAYS correspond the first index of the two-dimension arrays, and the vertical coordinates should ALWAYS be the second index. (Use transpose_3D_slice() to modify which variables are horizontal/vertical)

colors	The vector of colors used for rendering the false color tiles of the plot. If the string "default" is supplied instead, the default blue-white-red color scale is used, where the color is bluer the more negative the dependent variable, redder the more positive the dependent variable, and which white is anchored at 0.
NA_color	The color used to represent undefined or out-of-range values
contour_col	The color for the contour lines
xlabel	The label for the horizontal axis
ylabel	The label for the vertical axis
clabel	The label for the color bar. Defaults to c("Carbon", "(C/s)")
lwd	The linewidth of contour lines
...	Addtional arguments are passed to plot3D::image2D(), which is the "bottom" plot created by this function (to be overlaid by the contour plot)

Value

No explicit return (plot generated as side effect)

Examples

```
# create 3D slices
f <- function(x, y, z) { x + y * z }
g1 <- make_4D_grid(f, seq(1, 3, 0.1), seq(-2, 2, 0.2), seq(4, 10, 0.2))
s1 <- slice_4D_grid(g1, 2, 2)
# plot the 3D slice (if tcltk is available)
has_tk <- capabilities("tcltk") && (
  !grepl("darwin", R.version$os, ignore.case = TRUE) ||
  capabilities("X11")
)
if (has_tk) {
  plot_slice_3D(s1, contours=seq(5, 25, 2.5), dims=c(1, 3, 4))
}
```

---

print.DHScale *Customized print for S3 class 'DHScale'*

Description

Customized print for S3 class 'DHScale'

Usage

```
## S3 method for class 'DHScale'
print(x, unicode = TRUE, ...)
```

Arguments

x	the 'DHScale' instance to be printed.
unicode	whether to print using unicode characters.
...	the remaining arguments are ignored.

Value

No explicit return. Information of the 'DHScale' instance is printed to stdout.

Examples

```
ia_cyano_DHScale <- new_DHScale(40.1, 38.8, 26.7, -27.4, name = "Form Ia, cyanobacteria")
print(ia_cyano_DHScale) # printout DHScale info
```

---

print.enzyme *Customized print for S3 class 'enzyme'*

Description

Customized print for S3 class 'enzyme'

Usage

```
## S3 method for class 'enzyme'
print(x, unicode = TRUE, ...)
```

Arguments

x	the 'enzyme' instance to be printed.
unicode	whether to print using unicode characters.
...	the remaining arguments are ignored.

Value

No explicit return. Information of the enzyme instance is printed to stdout.

Examples

```
pro_enzyme <- new_enzyme(6.6, 144, 394, 10.8, name="pro") # new instance
print(pro_enzyme) # print out enzyme info
```

refs_to_citation *References table*

Description

A table containing a list of the citation information and pmid/doi numbers to help with citing data from the package.

Usage

```
refs_to_citation
```

Format

'refs_to_citation':

A data frame with 150 rows and 3 columns:

short_ref Short reference

pmid_or_doi The pmid number or the doi

citation Full citation

Source

compiled by the authors

Rubisco_25C *Comprehensive Rubisco kinetics data*

Description

A dataset of Rubisco kinetics data compiled from the literature. Includes all four major kinetic parameters (kcat, Kc, Ko, and Sc/o), along with ancillary information about the measurements and the studies.

Usage

```
Rubisco_25C
```

Format

'Rubisco_25C':

A data frame with 1454 rows and 30 columns:

identifier Unique entry ID

genus Genus

species Species

subspecies Subspecies, variant, or cultivar
mutant Species
mutant_details Species
primary Species
heterologous_expression Species
kcat_val Value of kcat (/sec)
kcat_T kcat measurement temperature (°C)
kcat_pH kcat measurement pH
Kc_val Value of Kc (μM)
Kc_T Kc measurement temperature (°C)
Kc_pH Kc measurement pH
Ko_val Value of Ko (μM)
Ko_T Ko measurement temperature (°C)
Ko_pH Ko measurement pH
S_val Value of Sc/o
S_T Specificity measurement temperature (°C)
S_pH Specificity measurement pH
temp Kinetics measurement temperature (°C)
pH Kinetics measurement pH
pKa pKa used to determine CO2 concentration for Kc, Ko, and specificity calculations
form Form of Rubisco
PGS Recommended phosphoglycolate salvage pathway
taxonomy Taxonomy
group Broader phylogenetic group
note Note
short_ref Short reference
Year Study year

Source

compiled by the authors from a literature search

`Rubisco_abridged` *Abridged Rubisco kinetics data*

Description

A subset of data from the comprehensive Rubisco kinetics table also included in the package, plus some composite data and median Rubisco kinetics as described in sections 5.1 and 5.2 of the Creating Custom Enzymes vignette.

Usage

```
Rubisco_abridged
```

Format

'Rubisco_abridged':
A data frame with 294 rows and 19 columns:

identifier Unique entry ID
genus Genus
species Species
subspecies Subspecies, variant, or cultivar
kcat_val Value of kcat (/sec)
kcat_T kcat measurement temperature (°C)
Kc_val Value of Kc (μM)
Kc_T kcat measurement temperature (°C)
Ko_val Value of Ko (μM)
Ko_T kcat measurement temperature (°C)
S_val Value of Sc/o
S_T Specificity measurement temperature (°C)
temp Kinetics measurement temperature (°C)
form Form of Rubisco
PGS Recommended phosphoglycolate salvage pathway
taxonomy Taxonomy
group Broader phylogenetic group
note Note
short_ref Short reference

Source

compiled by the authors from a literature search

`Rubisco_aliases` *Rubisco alias table*

Description

A table containing a list of some of the alternative names for species contained within the datasets. This includes both common names (i.e., "wheat") and former names that are no longer in use (i.e., "Chenopodium rubra", which has become Oxybasis rubra).

Usage

```
Rubisco_aliases
```

Format

'Rubisco_aliases':

A data frame with 79 rows and 4 columns:

alternate_name Alternate, user-input name

genus_database Genus name in database

species_database Species name in database

subspecies_database Subspecies name in database

Source

compiled by the authors

---

search_alias *Search for data of Rubisco enzyme alias from the database*

Description

Data is pulled either from the abridged table or the comprehensive table, both of which are included in this package

Usage

```
search_alias(string, data = NULL, match = "complete")
```

Arguments

string	a string to search for in the database
data	the source data to search from. Can be "abridged", "comprehensive", or NULL. If NULL, the search will proceed from abridged to comprehensive until a match is found
match	whether to require complete match. Can be "complete" or "partial"

Value

a dataframe containing matching entries.

search_DHScale *Search for temperature dependence DHScale from the database*

Description

Data is pulled either from the averaged table or the abridged table, both of which are included in this package

Usage

```
search_DHScale(string, level = NULL, data = NULL, match = "complete")
```

Arguments

string	a string to search for in the database
level	the level at which to search. Can be "species", "genus", "taxonomy", "form", "group", or NULL. If NULL, the search will proceed from genus to species to taxonomy to form to group until a match is found. Note that on the averaged table the genus is always "Average," while the species generally agrees with "form," except for the overall average for which the specie is defined to be "Rubisco"
data	the source data to search from. Can be "averaged", "abridged", or NULL. If NULL, the search will proceed from abridged to averaged until a match is found
match	whether to require complete match. Can be "complete" or "partial"

Value

a dataframe containing matching entries.

Examples

```
Aegilops <- search_DHScale("Aegilops", level="genus") # complete genus search
print(Aegilops) # printout search results
```

search_enzyme *Search for data of Rubisco enzyme from the database*

Description

Data is pulled either from the abridged table or the comprehensive table, both of which are included in this package

Usage

```
search_enzyme(string, level = NULL, data = NULL, match = "complete")
```

Arguments

string	a string to search for in the database (case insensitive)
level	the taxonomic level to search. Can be "alias", "genus", "species", "form", "taxonomy", "group" or NULL. If NULL, the search will proceed from alias to genus to species to taxonomy to form to group until a match is found
data	the source data to search from. Can be "abridged", "comprehensive", or NULL. If NULL, the search will proceed from abridged to comprehensive until a match is found
match	whether to require complete match. Can be "complete" or "partial"

Value

a dataframe containing matching entries.

Examples

```
Aegilops <- search_enzyme("Aegilops", level="genus") # complete genus search
print(Aegilops) # printout search results
```

---

slice_4D_grid *Create a 3D slice of 4D grid created by make_4D_grid().*

Description

The resulting slice remains a list of 4 elements, but each element except the dimension being sliced will be a 2-dimensional array. For the dimension being sliced, the element will be a scalar showing the constant value of the sliced coordinate.

Usage

```
slice_4D_grid(grid, dim, val, approx = TRUE, tol = 0.001)
```

Arguments

grid	The original grid to be sliced.
dim	The dimension to be sliced. Its corresponding coordinates will take constant values in the resulting slice.
val	The constant value that the dim-th coordinates will take in the resulting slice.
approx	Whether approximate value matching is allowed.
tol	the tolerance for approximate value matching.

Value

A 3D slice of the supplied 3D grid.

Examples

```
f <- function(x, y, z) { x + y * z }
g1 <- make_4D_grid(f, 1:3, c(-2,2), seq(4, 10, 2))
s1 <- slice_4D_grid(g1, 2, 2)
s1
```

`temp_dep_abridged` *Abridged Delta H temperature scaling data*

Description

A dataset of Rubisco temperature dependence data compiled from a literature search.

Usage

```
temp_dep_abridged
```

Format

'temp_dep_abridged':

A data frame with 182 rows and 13 columns:

identifier Unique entry ID

genus Genus

species Species

subspecies Subspecies, variant, or cultivar

kcat_dH kcat temperature dependence parameter (kJ/mol)

Kc_dH Kc temperature dependence parameter (kJ/mol)

Ko_dH Ko temperature dependence parameter (kJ/mol)

S_dH S temperature dependence parameter (kJ/mol)

form Form of Rubisco

taxonomy Taxonomy

group Broader phylogenetic group

note Note

short_ref Short reference

Source

compiled by the authors from a literature search

`temp_dep_averaged` *Averaged Delta H temperature scaling data*

Description

A dataset of average Rubisco temperature scaling data per taxonomic group and form, as described in Section 5.3 of the "Designing Custom Enzymes" vignette.

Usage

```
temp_dep_averaged
```

Format

'temp_dep_averaged':
A data frame with 9 rows and 13 columns:

identifier Unique entry ID
genus Genus
species Species
subspecies Subspecies, variant, or cultivar
kcat_dH kcat temperature dependence parameter (kJ/mol)
Kc_dH Kc temperature dependence parameter (kJ/mol)
Ko_dH Ko temperature dependence parameter (kJ/mol)
S_dH S temperature dependence parameter (kJ/mol)
form Form of Rubisco
taxonomy Taxonomy
group Broader phylogenetic group
note Note
short_ref Short reference

Source

compiled by the authors, mostly from Galmés et al. 2016

`transpose_3D_slice` *Transpose the non-constant independent variables in a 3D grid created by slice_4D_grid().*

Description

In the new slice, every non-constant element in the list is transposed. Moreover, the order of the 2 non-constant independent variables in the list is also swapped.

Usage

```
transpose_3D_slice(slice)
```

Arguments

slice	The original slice to be transposed.

Value

A transposed 3D slice.

Examples

```
f <- function(x, y, z) { x + y * z }
g1 <- make_4D_grid(f, 1:3, c(-2,2), seq(4, 10, 2))
s1 <- slice_4D_grid(g1, 2, 2)
s2 <- transpose_3D_slice(s1)
s2
```

Index