Package 'gasanalyzer'

December 4, 2024

Type Package

Title Import, Recompute and Analyze Data from Portable Gas Analyzers

Version 0.4.2

Date 2024-12-02

Description The gasanalyzer R package offers methods for importing, preprocessing, and analyzing data related to photosynthetic characteristics (gas exchange, chlorophyll fluorescence and isotope ratios). It translates variable names into a standard format, and can recalculate derived, physiological quantities using imported or predefined equations. The package also allows users to assess the sensitivity of their results to different assumptions used in the calculations.
See also Tholen (2024) <doi:10.1093/aobpla/plae035>.

License GPL-3

URL https://gitlab.com/plantphys/gasanalyzer

Depends R (>= 4.3.0)

Imports jsonify, methods, stats, stringi, tibble, tidyxl (>= 1.0.8), tools, units, utils, vctrs, xml2

Suggests knitr, rmarkdown, spelling, graphics, ggplot2, gridExtra, photosynthesis, testthat (>= 3.0.0)

VignetteBuilder knitr

Encoding UTF-8

Language en-US

RoxygenNote 7.3.2

Config/testthat/edition 3

NeedsCompilation no

Author Danny Tholen [aut, cre] (<https://orcid.org/0000-0002-9517-0939>, University of Natural Resources and Life Sciences, Vienna)

Maintainer Danny Tholen <thalecress+p@gmail.com>

Repository CRAN

Date/Publication 2024-12-03 23:10:12 UTC

Contents

`create_equations` *Create a list of equations for recalculating gasanalyzer data.*

Description

This function creates a list of equations that can be used to recalculate gas-exchange data by passing the resulting object to the `recalculate()` method. Various `useflags` can be defined to tune the equations. In addition, custom equations can be defined as arguments. Note that the calculations may fail if commons are missing in the gas-exchange data.

Usage

```
create_equations(useflags = "default", ...)
```

Arguments

`useflags`	character vector with the type of equations to create (such as c("li6800", "gfs3000")). Leave empty to obtain the default set. An unknown flag returns an empty list, and a warning listing all valid flags.
`...`	custom equations. the arguments must tagged function expressions. Note that the function body must be wrapped in curly brackets. The tags will be matched against the names of a data frame when applying the return value with `recalculate()`.

Value

A list of language objects with equations

See Also

read_6800_equations()

Examples

```
exampledir <- system.file("extdata", package = "gasanalyzer")

# import factory calibration for example data:
import_factory_cals(exampledir)

# read data from a txt file:
li6800 <- read_6800_txt(file.path(exampledir, "lowo2"))

# passing an invalid flags shows which flags are valid:
create_equations("help")

# create a default set of gas-exchange equations, for the 6800, but overwrite
# the default calculation of leaf light absorption with a custom value:
Eqs <- create_equations(c("default", "li6800"), LeafQ.alpha = \() {0.86})

#apply:
li6800_recalc <- recalculate(li6800, Eqs)

li6800$LeafQ.alpha
li6800_recalc$LeafQ.alpha
```

---

export_ess_dive *Export a subset of the data into the ESS-DIVE reporting format for leaf-level gas exchange data.*

Description

Ely et al. (2021) proposed a standardized nomenclature for reporting gas-exchange data and metadata within the framework of the Environmental System Science Data Infrastructure for a Virtual Ecosystem (ESS-DIVE) repository. This method converts data frames or tibbles created with `gasanalyzer` to this standardized format. Note that the scope of the proposed standard is limited, and therefore only a subset of the data is exported. Users should add relevant additional columns and provide relevant metadata.

Usage

```
export_ess_dive(df, filename = "", extra_cols = NULL)
```

Arguments

`df`	a tibble with gas-exchange data.
`filename`	path to the output file. If none provided the method returns the data as a tibble.
`extra_cols`	a character vector specifying additional columns (not specified in the standard specified by Ely et al. 2021) to include in the returned data or saved files.

Details

If a filename is given as argument, the data is written into a comma separated, UTF-8 encoded file without BOM and with CRLF line headings. In addition, a data dictionary file ("_dd" is inserted at the end of the filename, before the file extension). If no filename is provided, the converted data is returned.

Value

Nothing if a filename is provided. Otherwise, a tibble with variables and headings specified by the ESS-DIVE gas-exchange standard is returned.

References

Ely KS, Rogers A, Agarwal DA, et al (2021) *A reporting format for leaf-level gas exchange data and metadata.* Ecol Inform 61:101232. https://doi.org/10.1016/j.ecoinf.2021.101232

Examples

```
example <- system.file("extdata", "d13C.tsv", package = "gasanalyzer")

# read data and recalculate using default gas-exchange equations:
df <- read_gasexchange(example) |>
  recalculate(create_equations("default"))

# view df in ess_dive format:
export_ess_dive(df)


# save the data and a data dictionary:
export_ess_dive(df, "ess_dive_test.csv")
# read and show the dictionary:
readLines("ess_dive_test_dd.csv")
```

---

get_factory_cals *Returns a matrix with factory calibration information for given instrument serial numbers and calibration dates.*

Description

The factory calibration of the 6800 can be used to calculate concentrations from raw values. If calibration information is available in the package environment it can be retrieved by this method.

Usage

```
get_factory_cals(sn = NULL, datetime = NULL)
```

Arguments

sn	a character vector with an instrument serial number. If named, the names are kept in the output.
datetime	a POSIXct time vector indicating the latest possible time for the calibration data that is to be returned. If no calibration before datetime is found, the oldest available calibration is returned.

Details

The datetime option can be used to make sure that newer calibration files are not used in combination with older datafiles.

Value

A character matrix with factory calibration data. If no datetime is provided, the newest calibration is returned.

Examples

```
exampledir <- system.file("extdata", package = "gasanalyzer")

# import factory cals for example data:
import_factory_cals(exampledir)

# show calibration data for a specific instrument serial numbers, closest to
# the current time:
get_factory_cals(sn = "68H-422400", datetime=Sys.time())
```

---

import_factory_cals *Import instrument-specific factory calibration files from a folder.*

Description

The factory calibration of the 6800 can be used to calculate concentrations from raw values. The calibration files are found in subfolders of /home/licor/.factory on the instrument. They can be copied to a computer and imported to the package configuration files using this method.

Usage

```
import_factory_cals(
  folder = tools::R_user_dir("gasanalyzer", which = "config"),
  keep = FALSE
)
```

Arguments

folder	A folder where calibration files are to be found.
keep	Copies valid calibration files to a package-specific configuration folder. Will result in automatic import of the data in the future. Will overwrite files.

Details

The function will also load the calibration into the package environment, where they can be retrieved by `get_factory_cals()`.

This method assumes the files are named with serial number and calibration date, separated by an underscore.

Value

Calibration data is stored in the package environment.

See Also

`get_factory_cals()`

Examples

```
exampledir <- system.file("extdata", package = "gasanalyzer")

# show calibration data
get_factory_cals()

# import factory calibration for example data:
import_factory_cals(exampledir)

# show calibration data
get_factory_cals()
```

`modify_equations` *Modify an existing list of equations with specific user-specified equations.*

Description

This method allows replacing a specific equations in a list with custom versions. Although it is possible to add custom equations using `create_equations()`, it can be useful to modify existing sets. It can also be used to modify equations imported from an `xlsx` file.

Usage

```
modify_equations(eqs, ...)
```

Arguments

`eqs`	a list of calls for recomputing `gasanalyzer` equations.
`...`	custom equations. the arguments must tagged function expressions. The tags will be matched against the equation list specified in eqs, and matching expressions will be replaced. Additional expressions will be added to the list. Note that the function body must be wrapped in curly brackets.

Value

A modified list of calls containing equations to recalculate `gasanalyzer` data.

See Also

`read_6800_equations()`

Examples

```
exampledir <- system.file("extdata", package = "gasanalyzer")

# import factory calibration for example data:
import_factory_cals(exampledir)

# read data from a txt file:
li6800 <- read_6800_txt(file.path(exampledir, "lowo2"))

# create a default set of gas-exchange equations, for the Li-6800:
Eqs <- create_equations(c("default", "li6800"))

# replace the value for the leaf light absorptance:
Eqs <- modify_equations(Eqs, LeafQ.alpha = \() {0.86})

# apply:
li6800_recalc <- recalculate(li6800, Eqs)

li6800$LeafQ.alpha
li6800_recalc$LeafQ.alpha
```

---

permutate *Expand a data frame with all possible combinations of the values in a column.*

Description

For sensitivity analyses, it is useful to permutate the values in a single column, whilst keeping all other values constant. After creating such a permutation, `recalculate()` should be used to analyze the effect of the change in the column of interest. If the effect of changes in multiple columns is to be analyzed, this function can be called in series.

Usage

```
permutate(df, ...)
```

Arguments

`df`	a dataframe or tibble
`...`	a name-value pair. The name gives the name of the column in the input that is to be changed. The value is a vector specifying all values that are desired in the output. For every value in this vector, all other rows are duplicated.

Value

a data frame containing all possible combinations of the input df and the vector specified in ...

Note that the units and classes of the columns in the input data frame are applied to the replacement values. Unexpected behavior may occur when providing incompatible classes or units.

Examples

```
example <- system.file("extdata", "6400-testfile", package = "gasanalyzer")

# read data:
li6400 <- read_6400_txt(example)

# expand the data frame for a range of leaf areas, and recalculate the data:
li6400 <- permutate(li6400, Const.S = seq(1, 8)) |>
  recalculate(create_equations(c("default", "li6400")))

if (interactive()) {
  require(units)
  require(graphics)

  # observe that changing the leaf area enclosed in the chamber would have a
  # nonlinear effect on the rate of photosynthesis:
  aggregate(list(A = li6400$GasEx.A), list(Area = (li6400$Const.S)), mean) |>
    plot()
}
```

`read_6400_txt` *Reads 6400XT text files and creates a tibble with gas-exchange data*

Description

The text files stored by the 6400 contain measured and calculated values that are read by this function and formatted in a large tibble for use with R. Constants and metadata are also added as columns. Note that no recalculation of derived variables is performed, although it is possible to so using `recalculate()` after importing the data.

Usage

```
read_6400_txt(filename, tz = Sys.timezone())
```

Arguments

`filename`	an text file containing 6400XT gas-exchange data.
`tz`	a character string specifying the timezone for the loaded file. If omitted, the current time zone is used. Invalid values are typically treated as UTC, on some platforms with a warning.

Details

Multiple files can be loaded by calling the function with `lapply()` or `purrr::map()` to merge multiple files. In this case, it is important to ensure that the column names will match.

Value

A tibble with gas-exchange data in columns.

See Also

recalculate

Examples

```
example <- system.file("extdata", "6400-testfile", package = "gasanalyzer")

# read data
li6400data <- read_6400_txt(example)

#View
li6400data
```

---

read_6800_equations *Read gas-exchange equations directly from 6800 xlsx files.*

Description

It is recommended to use `create_equations()` for recalculating gas-exchange data, but under some conditions, it may be useful to apply exactly the same equations as used in the xlsx data file.

Usage

```
read_6800_equations(filename)
```

Arguments

`filename`	an xlsx file containing 6800 gas-exchange data

Details

Currently, this only works for xlsx files stored by the 6800. this function extracts xlsx formulas from the file and stores them in a list for use by the `recalculate()` function. Note there is no guarantee that the extracted equations work on any other data files. Since newer versions of the 6800 firmware allows defining custom equations, it is not guaranteed that all equations can be extracted successfully.

Value

A list of gas-equations.

In principle, this can be made to work for the 6400 as well, but since that instrument uses a variation of the older xls format, it is hard to get working in practice.

See Also

`create_equations()`

Examples

```
example <- system.file("extdata", "lowo2.xlsx", package = "gasanalyzer")

# get equations stored in the xlsx file
Eqs <- read_6800_equations(example)

#Inpect how stomatal conductance is calculated:
Eqs$GasEx.gsw
```

`read_6800_txt` *Reads 6800 text files and creates a tibble with gas-exchange data.*

Description

The text files stored by the 6800 contain measured and calculated values that are read by this function and formatted in a large tibble for use with R. Constants and metadata (such as calibration information) are also added as columns. Note that no recalculation of derived variables is performed, although it is possible to so using `recalculate()` after importing the data.

Usage

```
read_6800_txt(filename)
```

Arguments

`filename`	an text file containing 6800 gas-exchange data.

Details

Multiple files can be loaded by calling the function with `lapply()` or `purrr::map()` to merge multiple files. In this case, it is important to ensure that the column names will match.

Value

A tibble with gas-exchange data in columns.

See Also

`recalculate()`

Examples

```
exampledir <- system.file("extdata", package = "gasanalyzer")
# import factory calibration for example data:
import_factory_cals(exampledir)

# read data
li6800 <- read_6800_xlsx(file.path(exampledir, "lowo2.xlsx"))
li6800_txt <- read_6800_txt(file.path(exampledir, "/lowo2"))

# compare all except equations. Note txt file reports some NAs as zero:
columns_to_check <- names(li6800)[!names(li6800) %in%
                                    c("gasanalyzer.Equations")]
all.equal(li6800[columns_to_check],
          li6800_txt[columns_to_check],
          tol = 0.01)
```

`read_6800_xlsx` *Reads 6800 xlsx files and creates a tibble with gas-exchange data.*

Description

The xlsx files stored by the 6800 contain measured and calculated values that are read by this function and formatted in a large tibble for use with R. Constants and metadata (such as calibration information) are also added as columns.

Usage

```
read_6800_xlsx(filename, recalculate = TRUE)
```

Arguments

`filename`	an xlsx file containing 6800 gas-exchange data.
`recalculate`	character string indicating whether or not to recalculate data using equations from the xlsx file.

Details

Note that values for many derived gas-exchange parameters are not stored in the files, but are calculated by equations stored in the xlsx. These values are 0 after importing, unless setting recalculate = TRUE. It is also possible to calculate this parameters after importing using the `recalculate()` function.

Multiple files can be loaded by calling the function with `lapply()` or `purrr::map()` to merge multiple files. In this case, it is important to ensure that the column names will match. Recalculation can be disabled for speed, and instead applied to the merged data using `recalculate()`.

Value

A tibble with gas-exchange data in columns.

See Also

recalculate()

Examples

```
exampledir <- system.file("extdata", package = "gasanalyzer")
# import factory calibration for example data:
import_factory_cals(exampledir)

# read data:
li6800 <- read_6800_xlsx(file.path(exampledir, "lowo2.xlsx"))
li6800_norecalc <- read_6800_xlsx(file.path(exampledir, "lowo2.xlsx"),
  recalculate = FALSE)
li6800_norecalc$gasanalyzer.Equations <-
  list(read_6800_equations(file.path(exampledir, "lowo2.xlsx")))

all.equal(li6800, recalculate(li6800_norecalc), check.attributes = FALSE)
```

---

read_ciras4 *Reads CIRAS-4 csv files and creates a tibble with gas-exchange data*

Description

The csv files stored by the CIRAS-4 contain measured and calculated values that are read by this function and formatted in a large tibble for use with R. Note that no recalculation of derived variables are performed, although it is possible to do so using recalculate() after importing the data.

Usage

```
read_ciras4(filename)
```

Arguments

filename	a csv file containing gas-exchange data.

Details

Multiple files can be loaded by calling the function with lapply() or purrr::map() to merge multiple files. In this case, it is important to ensure that the column names will match.

Value

a tibble with gas-exchange data in columns.

See Also

recalculate()

Examples

```
example <- system.file("extdata", "ciras4.csv", package = "gasanalyzer")

# Read using unified column names:
cir4 <- read_ciras4(example)

# Recalculate data using default gas exchange equations:
cir4_recalc <- recalculate(cir4, create_equations(c("default", "ciras4")))

# View differences:
all.equal(cir4, cir4_recalc[names(cir4)], tol = 0.001)
```

---

read_gasexchange *Read gas-exchange data from a text file.*

Description

Data stored by write_gasexchange() can be read by this method. The first row is the header, the second row specify the units. File encoding must be UTF-16LE (use the export as unicode txt option in Microsoft Excel).

Usage

```
read_gasexchange(filename, delim = "\t")
```

Arguments

filename	path to the input file
delim	delimiter to use for the file

Value

a tibble with gas-exchange data

Examples

```
example <- system.file("extdata", "d13C.tsv", package = "gasanalyzer")

# read data
read_gasexchange(example)
```

read_gfs *Reads GFS-3000 text files and creates a tibble with gas-exchange data*

Description

The text files stored by the GFS-3000 contain measured and calculated values that are read by this function and formatted in a large tibble for use with R. Note that no recalculation of derived variables is performed, although it is possible to do so using `recalculate()` after importing the data.

Usage

```
read_gfs(
  filename,
  tz = Sys.timezone(),
  unified_names = TRUE,
  skip_to_data = 2,
  delim = ";"
)
```

Arguments

`filename`	an xlsx file containing 6800 gas-exchange data.
`tz`	a character string specifying the timezone for the loaded file. If omitted, the current time zone is used. Invalid values are typically treated as UTC, on some platforms with a warning.
`unified_names`	= TRUE, use unified column names. This is necessary for further processing of the data using this package.
`skip_to_data`	use skip=4 if the file has a double header.
`delim`	= ";" Allows specified the delimiter used in the files. Re-saved data may use a comma as delimiter.

Details

Multiple files can be loaded by calling the function with `lapply()` or `purrr::map()` to merge multiple files. In this case, it is important to ensure that the column names will match.

Value

a tibble with gas-exchange data in columns and equations as attribute.

See Also

`recalculate()`

Examples

```
example <- system.file("extdata", "aci1.csv", package = "gasanalyzer")

# Read using GFS-3000 names and formatting:
gfs3000_old <- read_gfs(example, unified_names = FALSE)
# Read using unified column names:
gfs3000 <- read_gfs(example)

# Inspect the intercellular CO2:
gfs3000_old$ci
gfs3000$GasEx.Ci

# Recalculate data using default gas exchange equations:
gfs3000 <- recalculate(gfs3000, create_equations(c("default", "gfs3000")))
gfs3000$GasEx.Ci
```

---

recalculate *Recalculate gas-exchange data based on a set of equations.*

Description

The recalculation uses equations in a list of quosures provided as argument. This list can be obtained from `create_equations()` or `read_6800_equations()`.

Usage

```
recalculate(df, eqs = NULL)
```

Arguments

df	A data frame or an extension thereof (e.g. a tibble).
eqs	a list of quosures that define how the df will be altered.

Value

A tibble with recalculated columns as specified by the eqs argument

Examples

```
exampledir <- system.file("extdata", package = "gasanalyzer")
# import factory calibration for example data:
import_factory_cals(exampledir)

# read data:
li6800 <- read_6800_xlsx(file.path(exampledir, "lowo2.xlsx"))

# recalculate using xlsx equations:
li6800 <- recalculate(li6800)
```

```
# recalculate using gasanalyzer default equations for the li6800:
li6800_ge <- recalculate(li6800, create_equations(c("default", "li6800")))

# the difference is that units have been enforced using gasanalyzer, which
# has been recorded in a column:
all.equal(li6800, li6800_ge[names(li6800)], tol = 0.01)
```

---

var2label *Render gasanalyzer variables or values using mathematical notation.*

Description

The argument is converted to a plotmath expression that can be used using `plot` or `ggplot2`. If there is no known plotmath expression defined, the argument is returned as is.

Usage

```
var2label(varname, use_units = FALSE, val = NULL, ...)
```

Arguments

`varname`	a character list or vector argument with variable names.
`use_units`	whether or not to append default units to the resulting. expression. It is better to rely on the units in the actual data, which is handled automatically by newer ggplot2 versions.
`val`	a value to display rather than the variable name. Needs to be of the same length as varname and if not a character vector it should be coercible to one. NAs are replaced with the plotmath expressions.
`...`	options passed on to make_unit_label.

Value

a list of plotmath expressions.

See Also

plotmath

Examples

```
# make labels
lbls <- var2label(c("GasEx.Ci", "GasEx.A"), use_units = TRUE)
print(lbls)

# plot
plot(1, type = "n", xlab = lbls[[1]], ylab = lbls[[2]])
# add temperature as title, removing [] from the units:
title(main = var2label("Meas.Tleaf",use_units = TRUE, val = 25,
  group = c("", ""))[[1]])
```

`write_gasexchange` *Write a gas-exchange tibble to a text file.*

Description

The column names and column units are saved as a two-row header. The files use UTF-16LE and CRLF line headings for compatibility. By default, tabs are used as delimiter. If you intend to open the file in a spreadsheet program, it may be helpful to use `csv` as file extension.

Usage

```
write_gasexchange(df, filename, delim = "\t")
```

Arguments

`df`	a tibble with gas-exchange data
`filename`	path to the output file
`delim`	delimiter to use for the file

Details

Note that for data-exchange between R sessions, `saveRDS()` and `readRDS()` are faster, and support saving and loading list columns (such as calibration information and equations). This method is primarily meant for exchanging data with other software packages.

Value

No return value. If there a problems writing the file, a warning or error will be shown.

Examples

```
example <- system.file("extdata", "d13C.tsv", package = "gasanalyzer")

# read data and recalculate using default gas-exchange equations:
df <- read_gasexchange(example) |>
  recalculate(create_equations("default"))

# write recaculated data
write_gasexchange(df, "d13C_recalculated.tsv")
```

Index