

Package ‘gasanalyzer’

January 29, 2025

Type Package

Title Import, Recompute and Analyze Data from Portable Gas Analyzers

Version 0.4.3

Date 2025-01-28

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](https://doi.org/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, nleqslv

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 2025-01-29 10:20:06 UTC

Contents

create_equations	2
export_ess_dive	6
get_factory_cals	7
import_factory_cals	8
modify_equations	9
permutate	10
read_6400_txt	11
read_6800_equations	12
read_6800_txt	13
read_6800_xlsx	14
read_ciras4	15
read_gasexchange	16
read_gfs	17
recalculate	18
var2label	19
write_gasexchange	20

Index	21
--------------	-----------

create_equations	<i>Create a list of equations for recalculating gasanalyzer data.</i>
------------------	---

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 (see Details). 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 <code>recalculate()</code> .

Details

The `useflags` argument currently supports several pre-defined sets of equations.

Gas-Exchange and fluorescence models:

`default` A set with the most commonly-used derived quantities in the gas-exchange (GasEx) and chlorophyll fluorescence (FLR) categories. The equations are described in detail by von Caemmerer and Farquhar (1981) and LI-COR Biosciences Inc (2022).

`GoffGratch1946` Replaces or adds equations related to the calculation of the saturated water pressure of the leaf (GasEx.SVPleaf) and chamber air (GasEx.SVPcham). The calculation is based on that described by Goff and Gratch (1946) and only takes the temperature into account.

`Buck1981` Replaces or adds equations for (GasEx.SVPleaf) and (GasEx.SVPcham). based on the description by Buck (1981). Takes temperature and pressure into account.

`Buck1996` Replaces or adds equations for (GasEx.SVPleaf) and (GasEx.SVPcham) based on the description by Buck (1996). Takes temperature and pressure into account.

`cuticular_conductance` Replaces the equations related to CO₂ and H₂O conductance and substomatal CO₂ (GasEx.gtw; GasEx.gsw; GasEx.gtc; GasEx.Ci) with versions that take into account cuticular conductance (Márquez et al., 2021, 2023). Requires manually specifying the cuticular conductance to water (Const.gcw) and CO₂ (Const.gcc).

`boundary_conducance` Replaces or adds equations related to the boundary layer conductance (GasEx.gbw) and leaf temperature derived from the energy balance equations (GasEx.TleafEB) with a version that is valid for a sample with no stomata. This is typically used when estimating GasEx.gbw using filter paper (Parkinson, 1985). Note that the leaf thermocouple (Meas.Tleaf) is used to estimate air temperature (and should therefore not touch the sample). This can be overridden by a custom equation. The sample emissivity (Const.eps), the ratio between heat and water conductance for the chamber air (Const.ra_rv) and the fraction of the area exchanging radiative and sensible heat (Const.asH) can be adjusted. For solving the energy balance equations, the method described by Bristow (1987) is used.

`gm_fluorescence` Adds derived variables for mesophyll conductance (FLR.gm) and chloroplast CO₂ mole fractions (FLR.Cc) based on gas exchange and chlorophyll fluorescence (Harley et al., 1992). It is strongly recommended to first calibrate the electron transport rates reported in the FLR.ETR column. In addition, the equations require adding columns for the respiration rate in the light (Const.RL), and the CO₂ photo-compensation point (Const.GammaStar).

Isotope models:

`d13C` Adds derived variables related to the stable carbon isotope discrimination model for C₃ plants (Farquhar and Cernusak, 2012; Evans and von Caemmerer, 2013) aimed at estimating the mesophyll conductance `d13C.gm` and chloroplast CO₂ mole fraction (`d13C.Cc`). Requires additional columns with data on the carbon isotope composition in sample and reference air (`d13CMeas.delta13CO2s`; `d13CMeas.delta13CO2r`) and in the air where the plants were grown (`d13CConst.delta13CO2g`), and values for respiration in the light (Const.RL) and the CO₂ photo-compensation point (Const.GammaStar).

`d13C_dis` Requires the `d13C` flag, but modifies the modeled carbon isotope discrimination `d13C.Delta` and `d13C.gm` using the assumption that the carbon pools of respiration and assimilation are disconnected (as described by Busch et al., 2020).

`d13C_e_Busch2020` Requires the `d13C` flag, but modifies the calculation of the effective respiratory fractionation (`d13C.ep`) to better take into account the effect of the growth conditions

(Busch et al. 2020). Additionally requires a value for the observed discrimination against $^{13}\text{CO}_2$ under growth conditions (`d13CConst.DeltaTag`).

Instrument related calculations:

These implement calculations as close as possible to those used in the instrument firmware and are typically related to the specific configurations or instrument design. In addition, some methods are provided for recalculating low-level instrument variables.

`li6400` Definitions for boundary layer conductance, temperature, and light related variables (`GasEx.gbw`; `GasEx.Rabs`; `GasEx.TairCnd`; `LeafQ.alpha`) specific to LI-6400 / LI-6400XT instruments (LI-COR Biosciences Inc, 2011).

`li6800` Definitions for boundary layer conductance, light, leakage, temperature and light related variables (`GasEx.gbw`; `GasEx.Rabs`; `GasEx.TairCnd`; `LeafQ.alpha`; `LeafQ.Qin`; `LeafQ.Conv`; `Leak.Fan`; `Leak.CorrFact`; `GasEx.Ca`) specific to the LI-6800 instrument (LI-COR Biosciences Inc, 2022).

`ciras4` Light absorptance (`LeafQ.alpha`) and the conversion between photon flux density and irradiance (`LeafQ.Conv`) is calculated by taking into account the effect of the different light sources that can be used with the CIRAS-4 instrument (PP Systems, 2024).

`gfs3000` Boundary layer conductance (`GasEx.gbw`) and light sensor (`LeafQ.Qin`) calculations adjusted for the default chamber of the GFS-3000 instrument (Heinz Walz GmbH, 2019).

`gfs3000_light_bot` Requires `gfs3000`, but modifies `LeafQ.Qin` to indicate that the bottom light sensor of the default chamber was used to quantify the light intensity incident on the leaf (Heinz Walz GmbH, 2019).

`match` Takes a previously stored offset between sample and reference analyzers into account when recalculating water and CO_2 mole fractions (`Meas.H2Os`; `Meas.CO2s`) as described by LI-COR Biosciences Inc (2022), Heinz Walz GmbH (2019) and PP Systems (2024). The corrected mole fractions are already stored in the data and therefore this calculation is usually not required. However, these equations are needed when recalculating mole fractions from lower level data (see the `raw` and `O2_correction` flags).

`raw` Recalculates CO_2 and H_2O mole fractions from such low-level variables (`Meas.CO2r`; `Meas.H2Or`; `Meas.CO2a`; `Meas.H20a`) as described by LI-COR Biosciences Inc (2022). Currently only implemented for the LI-6800 because low-level instrument data are required. Requires storing raw data and the availability of factory calibration files. Requires and enables the `match` set.

`O2_correction` Recalculates CO_2 and H_2O mole fractions (`Meas.CO2r`; `Meas.H2Or`; `Meas.CO2a`; `Meas.H20a`) at a potentially different oxygen concentration (`Const.Oxygen`). Currently only implemented for the LI-6800 (LI-COR Biosciences Inc, 2022) and GFS-3000 (K. Siebke, Heinz Walz GmbH, personal communication). For the LI-6800, this requires loading of factory calibration files (`import_factory_cals()`). Requires and automatically enables the `match` flag.

Value

A list of language objects with equations

References

- Bristow KL. 1987. On solving the surface energy balance equation for surface temperature. *Agricultural and Forest Meteorology* **39**:49–54.

- Buck AL. 1981. New equations for computing vapor pressure and enhancement factor. *Journal of Applied Meteorology* **20**:1527–1532.
- Buck AL. 1996. Buck research CR-1A user's manual. Boulder, CO: Buck Research Instruments.
- Busch FA, Holloway-Phillips M, Stuart-Williams H, Farquhar GD. 2020. Revisiting carbon isotope discrimination in C3 plants shows respiration rules when photosynthesis is low. *Nature Plants* **6**:245–258. <https://www.hygrometers.com/wp-content/uploads/CR-1A-users-manual-2009-12.pdf>
- von Caemmerer S, Farquhar GD. 1981. Some relationships between the biochemistry of photosynthesis and the gas exchange of leaves. *Planta* **153**:376–387.
- Evans JR, von Caemmerer S. 2013. Temperature response of carbon isotope discrimination and mesophyll conductance in tobacco. *Plant, Cell & Environment* **36**:745–756.
- Farquhar GD, Cernusak LA. 2012. Ternary effects on the gas exchange of isotopologues of carbon dioxide. *Plant, Cell & Environment* **35**:1221–1231.
- Goff JA, Gratch S. 1946. Low-pressure properties of water from -160° F to 212° F. *Transactions of the American Society of Heating and Ventilating Engineers* **52**:95–121.
- Harley PC, Loreto F, Di Marco G, Sharkey TD. 1992. Theoretical considerations when estimating the mesophyll conductance to CO₂ flux by analysis of the response of photosynthesis to CO₂. *Plant Physiology* **98**:1429–1436.
- Heinz Walz GmbH. 2019. Portable gas exchange fluorescence system GFS-3000. Handbook of operation. 9th edition. https://www.walz.com/files/downloads/manuals/gfs-3000/GFS-3000_Manual_9.pdf
- LI-COR Biosciences Inc. 2011. Using the LI-6400/LI-6400XT portable photosynthesis system. Version 6.2. <https://www.licor.com/env/support/LI-6400/manuals.html>
- LI-COR Biosciences Inc. 2022. Using the LI-6800 portable photosynthesis system. Version 2.1. <https://www.licor.com/env/support/LI-6800/manuals.html>
- Márquez DA, Stuart-Williams H, Farquhar GD. 2021. An improved theory for calculating leaf gas exchange more precisely accounting for small fluxes. *Nature Plants* **7**:317–326.
- Márquez DA, Stuart-Williams H, Cernusak LA, Farquhar GD. 2023. Assessing the CO₂ concentration at the surface of photosynthetic mesophyll cells. *New Phytologist* **238**:1446–1460.
- Parkinson KJ. 1985. A simple method for determining the boundary layer resistance in leaf cuvettes. *Plant, Cell & Environment* **8**: 223–226.
- PP Systems. 2024. CIRAS-4 portable photosynthesis system. Operation manual. Amesbury, MA: PP Systems. Version 1.3.

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(exampdir, "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	<i>Export a subset of the data into the ESS-DIVE reporting format for leaf-level gas exchange data.</i>
-----------------	---

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

<code>df</code>	a tibble with gas-exchange data.
<code>filename</code>	path to the output file. If none provided the method returns the data as a tibble.
<code>extra_cols</code>	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	<i>Returns a matrix with factory calibration information for given instrument serial numbers and calibration dates.</i>
------------------	---

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

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

# import factory calcs for example data:
import_factory_cals(exampdir)

# 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

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

# show calibration data
get_factory_cals()

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

# show calibration data
get_factory_cals()
```

modify_equations	<i>Modify an existing list of equations with specific user-specified equations.</i>
------------------	---

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 permute 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)

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

read_6800_txt	<i>Reads 6800 text files and creates a tibble with gas-exchange data.</i>
---------------	---

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	<i>Reads 6800 xlsx files and creates a tibble with gas-exchange data.</i>
----------------	---

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	<i>Read gas-exchange data from a text file.</i>
------------------	---

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	<i>Reads GFS-3000 text files and creates a tibble with gas-exchange data</i>
----------	--

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

<code>filename</code>	an xlsx file containing 6800 gas-exchange data.
<code>tz</code>	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.
<code>unified_names</code>	= TRUE, use unified column names. This is necessary for further processing of the data using this package.
<code>skip_to_data</code>	use skip=4 if the file has a double header.
<code>delim</code>	= ";" 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 are 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 recalculated data
write_gasexchange(df, "d13C_recalculated.tsv")
```

Index

create_equations, 2
create_equations(), 9, 12, 13, 18

export_ess_dive, 6

get_factory_cals, 7
get_factory_cals(), 8, 9

import_factory_cals, 8
import_factory_cals(), 4

lapply(), 12–15, 17

modify_equations, 9

permutate, 10
purrr::map(), 12–15, 17

read_6400_txt, 11
read_6800_equations, 12
read_6800_equations(), 5, 10, 18
read_6800_txt, 13
read_6800_xlsx, 14
read_ciras4, 15
read_gasexchange, 16
read_gfs, 17
readRDS(), 20
recalculate, 18
recalculate(), 2, 10–17

saveRDS(), 20

var2label, 19

write_gasexchange, 20
write_gasexchange(), 16