Package 'fluxible'

February 5, 2026

Title Ecosystem Gas Fluxes Calculations for Closed Loop Chamber Setup

Version 1.3.6

Date 2026-02-05

Description Toolbox to process raw data from closed loop flux chamber (or tent) setups into ecosystem gas fluxes usable for analysis. It goes from a data frame of gas concentration over time (which can contain several measurements) and a meta data file indicating which measurement was done when, to a data frame of ecosystem gas fluxes including quality diagnostics. Organized with one function per step, maximizing user flexibility and backwards compatibility. Different models to estimate the fluxes from the raw data are available: exponential as described in Zhao et al (2018) <doi:10.1016/j.agrformet.2018.08.022>, exponential as described in Hutchinson and Mosier (1981) <doi:10.2136/sssaj1981.03615995004500020017x>, quadratic, and linear. Other functions include quality assessment, plotting for visual check, calculation of fluxes based on the setup specific parameters (chamber size, plot area, ...), gross primary production and transpiration rate calculation, and light response curves.

License GPL (>= 3)

Encoding UTF-8

RoxygenNote 7.3.3

Suggests knitr, rmarkdown, testthat (>= 3.0.0), vdiffr, tidyverse, fs, licoread, readr

Config/testthat/edition 3

Imports broom, dplyr (>= 1.0.10), ggforce, ggplot2, haven, lubridate, rlang (>= 0.4.0), purrr, stats, stringr, tidyr, zoo, progress, purrrlyr, tidyselect, lifecycle, forcats, tibble

Depends R (>= 4.1)

LazyData true

URL https://plant-functional-trait-course.github.io/fluxible/, https://github.com/Plant-Functional-Trait-Course/fluxible

VignetteBuilder knitr

BugReports https://github.com/Plant-Functional-Trait-Course/fluxible/issues

NeedsCompilation no

Author Joseph Gaudard [aut, cre] (ORCID:
<https://orcid.org/0000-0002-6989-7624>),
Richard James Telford [aut] (ORCID:
<https://orcid.org/0000-0001-9826-3076>)

Maintainer Joseph Gaudard <joseph.gaudard@pm.me>

Repository CRAN

Date/Publication 2026-02-05 15:10:02 UTC

Contents

`co2_conc` *CO2 concentration*

Description

CO2 concentration with measurements meta data

Usage

```
co2_conc
```

Format

A tibble with 1251 rows and 13 variables

datetime Datetime at which CO2 concentration was recorded.

temp_air Air temperature inside the flux chamber in Celsius.

temp_soil Ground temperature inside the flux chamber in Celsius.

conc CO2 concentration in ppm.

PAR Photosynthetically active radiation inside the chamber in micromol/s/sqm.

turfID Unique ID of the turf in which the measurement took place.

type Type of measurement: ecosystems respiration (ER) or net ecosystem exchange (NEE).

f_start Datetime at which the measurement was started.

f_end Datetime at which the measurement ended.

f_fluxid Unique ID for each flux.

f_n_conc Number of data point per flux.

f_ratio Ratio of n_conc over length of the measurement (in seconds).

f_flag_match Data quality flags.

Examples

```
co2_conc
```

---

co2_df_short *CO2 concentration*

Description

Continuous CO2 concentration as measured on the field

Usage

```
co2_df_short
```

Format

A tibble with 1801 rows and 5 variables

datetime Datetime at which CO2 concentration was recorded.

temp_air Air temperature inside the flux chamber in Celsius.

temp_soil Ground temperature inside the flux chamber in Celsius.

conc CO2 concentration in ppm.

PAR Photosynthetically active radiation inside the chamber in micromol/s/sqm.

Examples

```
co2_df_short
```

`co2_fluxes` *CO2 fluxes*

Description

Manually calculated CO2 fluxes for testing purpose. df_short and record_short were used, with a zhao18 fit.

Usage

```
co2_fluxes
```

Format

A tibble with 6 rows and 11 variables

f_fluxid Unique ID for each flux.

f_slope_tz Slope of C(t) at t zero.

f_temp_air_ave Air temperature inside the flux chamber in Celsius averaged over the flux measurement.

f_flux CO2 flux in mmol/sqm/hour.

PAR Photosynthetically active radiation inside the chamber in micromol/s/sqm averaged over the flux measurement.

temp_soil Ground temperature inside the flux chamber in Celsius averaged over the flux measurement.

turfID Unique ID of the turf in which the measurement took place.

type Type of measurement: ecosystems respiration (ER) or net ecosystem exchange (NEE).

f_start Datetime at which the measurement started.

temp_fahr Air temperature inside the flux chamber in Fahrenheit averaged over the flux measurement.

temp_kelvin Air temperature inside the flux chamber in Kelvin averaged over the flux measurement.

Examples

```
co2_fluxes
```

co2_fluxes_lrc *CO2 fluxes with PAR values*

Description

CO2 fluxes with photosynthetically active radiation (PAR) for testing and examples of light response curves (LRC)

Usage

```
co2_fluxes_lrc
```

Format

A tibble with 257 rows and 5 variables

f_flux CO2 flux in mmol/sqm/hour.

datetime Time and date of the measurement.

PAR_ave Photosynthetically active radiation inside the chamber in micromol/s/sqm averaged over the flux measurement.

type Type of measurement: ecosystems respiration (ER), net ecosystem exchange (NEE), or light response curve (LRC).

warming Treatment: control or warming.

Examples

```
co2_fluxes_lrc
```

co2_liahovden *CO2 concentration at Liahovden*

Description

CO2 concentration at Liahovden site, used in example in readme file

Usage

```
co2_liahovden
```

Format

A tibble with 89692 rows and 5 variables

datetime Datetime at which CO2 concentration was recorded.

temp_air Air temperature inside the flux chamber in Celsius.

temp_soil Ground temperature inside the flux chamber in Celsius.

conc CO2 concentration in ppm.

PAR Photosynthetically active radiation inside the chamber in micromol/s/sqm.

Examples

```
co2_liahovden
```

`flux_calc` *Calculates ecosystem gas fluxes*

Description

Calculates a flux based on the rate of change of gas concentration over time

Usage

```
flux_calc(
  slopes_df,
  slope_col,
  f_datetime = f_datetime,
  temp_air_col,
  chamber_volume = deprecated(),
  setup_volume,
  atm_pressure,
  plot_area,
  f_fluxid = f_fluxid,
  conc_unit,
  flux_unit,
  cols_keep = c(),
  cols_ave = c(),
  cols_sum = c(),
  cols_med = c(),
  cols_nest = "none",
  tube_volume = deprecated(),
  temp_air_unit = "celsius",
  f_cut = f_cut,
  keep_arg = "keep",
  cut = TRUE,
  fit_type = c()
)
```

Arguments

`slopes_df`	dataframe of flux slopes
`slope_col`	column containing the slope to calculate the flux
`f_datetime`	column containing the datetime of each gas concentration measurements in `slopes_df`. The first one after cutting will be kept as datetime of each flux in the output.
`temp_air_col`	column containing the air temperature used to calculate fluxes. Will be averaged with NA removed.
`chamber_volume`	**[Deprecated]** see `setup_volume`

setup_volume	volume of the flux chamber and instrument together in L, can also be a column in case it is a variable
atm_pressure	atmospheric pressure in atm, can be a constant (numerical) or a variable (column name)
plot_area	area of the plot in m^2, can also be a column in case it is a variable
f_fluxid	column containing the flux IDs
conc_unit	unit in which the concentration of gas was measured mmol/mol, ppm, ppb, or ppt
flux_unit	desired units for the calculated fluxes. Has to be of the form amount/surface/time. Amount can be mol, mmol, umol, nmol or pmol. Time can be d (day), h (hour), mn (minute) or s (seconds). Surface can be m2, dm2 or cm2.
cols_keep	columns to keep from the input to the output. Those columns need to have unique values for each flux, as distinct is applied.
cols_ave	columns with values that should be averaged for each flux in the output. Note that NA are removed in mean calculation. Those columns will get the _ave suffix in the output.
cols_sum	columns with values for which is sum is provided for each flux in the output. Those columns will get the _sum suffix in the output.
cols_med	columns with values for which is median is provided for each flux in the output. Note that NA are removed in median calculation. Those columns will get the _med suffix in the output.
cols_nest	columns to nest in nested_variables for each flux in the output. Can be character vector of column names, "none" (default) selects none, or "all" selects all the column except those in cols_keep.
tube_volume	**[Deprecated]** see setup_volume
temp_air_unit	units in which air temperature was measured. Has to be either celsius (default), fahrenheit or kelvin.
f_cut	column containing cutting information
keep_arg	name in f_cut of data to keep
cut	if 'TRUE' (default), the measurements will be cut according to 'f_cut' before calculating fluxes. This has no influence on the flux itself since the slope is provided from flux_fitting, but it will influence the values of the variables in cols_ave, cols_cum, and cols_med.
fit_type	(optional) model used in flux_fitting. Will be automatically filled if slopes_df was produced using flux_fitting.

Value

a dataframe containing flux IDs, datetime of measurements' starts, fluxes (f_flux) in the units defined with flux_unit, temperature average for each flux in the same unit as the input (f_temp_ave), the model used in flux_fitting, any column specified in cols_keep, any column specified in cols_ave, cols_med or cols_sum with their values treated accordingly over the measurement after cuts, and a column nested_variables with the variables specified in cols_nest.

Examples

```
data(co2_conc)
slopes <- flux_fitting(co2_conc, conc, datetime, fit_type = "exp_zhao18")
flux_calc(slopes,
f_slope,
datetime,
temp_air,
conc_unit = "ppm",
flux_unit = "mmol/m2/h",
setup_volume = 24.575,
atm_pressure = 1,
plot_area = 0.0625)
```

`flux_diff` *Calculates difference between fluxes*

Description

to calculate a flux such as gross ecosystem production (GPP) or transpiration (T) as the difference between other fluxes (such as GPP = NEE - ER). Datetime and other variables to keep will be taken from the `type1` measurement. Fluxes not used here (soilR, LRC or other) are not lost.

Usage

```
flux_diff(
  fluxes_df,
  type_col,
  f_flux = f_flux,
  id_cols,
  type_a,
  type_b,
  diff_name,
  cols_keep = "none"
)
```

Arguments

`fluxes_df`	a dataframe containing fluxes
`type_col`	column containing type of flux
`f_flux`	column containing flux values
`id_cols`	columns used to identify each pair of fluxes
`type_a`	argument designating type_a fluxes in type column
`type_b`	argument designating type_b fluxes in type column
`diff_name`	name to give to the new calculated flux
`cols_keep`	columns to keep from `fluxes_df`. Values from type_a row will be filled in diff row. `none` (default) keeps only the columns in `id_cols`, flux, type and datetime columns; `all` keeps all the columns; can also be a vector of column names.

Value

a dataframe with $diff = type_a - type_b$ in long format with diff, type_a, and type_b as flux type, datetime, and any column specified in `cols_keep`. Values of datetime and columns in `cols_keep` for diff row are taken from type_a measurements.

Examples

```
data(co2_fluxes)
flux_diff(co2_fluxes, type, id_cols = "turfID", cols_keep = c("temp_soil"),
type_a = "NEE", type_b = "ER", diff_name = "GPP")
```

`flux_drygas` *wet air correction*

Description

Corrects for the amount of water vapor inside the air

Usage

```
flux_drygas(conc_df, gas_wet, h2o_wet)
```

Arguments

`conc_df`	dataframe of gas concentration over time
`gas_wet`	the gas to correct
`h2o_wet`	water vapor concentration before correction (in mmol/mol)

Details

the correction is done as follows $gas_dry = gas_wet/(1 - (h2o_wet/1000$

Value

the same dataframe with the additional column `[gas_wet]_dry` in the same unit as `gas_wet`

Examples

```
data(wet_conc)
flux_drygas(wet_conc, co2, h2o)
```

`flux_fitting` *Fitting a model to concentration data and estimating the slope*

Description

Fits gas concentration over time data with a model (exponential, quadratic or linear) and provides the slope later used to calculate gas fluxes with flux_calc

Usage

```
flux_fitting(
  conc_df,
  f_conc = f_conc,
  f_datetime = f_datetime,
  f_start = f_start,
  f_end = f_end,
  f_fluxid = f_fluxid,
  fit_type = "exp_zhao18",
  start_cut = 0,
  end_cut = 0,
  t_zero = 0,
  cut_direction = "none",
  cz_window = 15,
  b_window = 10,
  a_window = 10,
  roll_width = 15
)
```

Arguments

`conc_df`	dataframe of gas concentration over time
`f_conc`	column with gas concentration data
`f_datetime`	column with datetime of each concentration measurement Note that if there are duplicated datetime in the same `f_fluxid` only the first row will be kept
`f_start`	column with datetime when the measurement started (`ymd_hms`)
`f_end`	column with datetime when the measurement ended (`ymd_hms`)
`f_fluxid`	column with ID of each flux
`fit_type`	`exp_zhao18`, `exp_tz`, `exp_hm`, `quadratic` or `linear`. `exp_zhao18` is using the exponential model $C(t) = C_m + a(t - t_z) + (C_z - C_m)\exp(-b(t - t_z))$ from Zhao et al (2018). `expt_tz` is a modified version which allows the user to fix `t_zero`: $C(t) = C_m + a * t + (C_z - C_m)\exp(-b * t)$ `exp_hm` is using the HM model (Pedersen et al., 2010; Hutchinson and Mosier, 1981) $C(t) = C_m + (C_z - C_m)\exp(-b * t)$ `exponential` is equal to `exp_zhao18`, for backwards compatibility
`start_cut`	time to discard at the start of the measurements (in seconds)

end_cut	time to discard at the end of the measurements (in seconds)
t_zero	time at which the slope should be calculated (for quadratic, exp_tz and exp_hm fits)
cut_direction	"none" (default) means that the focus window is f_start + start_cut to f_end - end_cut; "from_start" makes it f_start + start_cut to f_start + end_cut; "from_end" makes it f_end - start_cut to f_end - end_cut. Bug fix since v1.3.4: "from_start" was doing f_start + start_cut to f_start + start_cut + end_cut
cz_window	window used to calculate Cz, at the beginning of cut window (exp_zhao18 and exp_tz fits)
b_window	window to estimate b. It is an interval after tz where it is assumed that the model fits the data perfectly (exp_zhao18 and exp_tz fits)
a_window	window at the end of the flux to estimate a (exp_zhao18 and exp_tz fits)
roll_width	width of the rolling mean for gas concentration when looking for tz, ideally same as cz_window (exp_zhao18 and exp_tz fits)

Value

a dataframe with the slope at t zero (f_slope), a datetime column of t zero (f_start_z), a factor column indicating the cuts (f_cut), the time in seconds since the start of the measurement (f_time), the modeled fit (f_fit), the modeled slope (f_fit_slope), the parameters of the fit depending on the model used, and any columns present in the input. The type of fit is added as an attribute for use by the other functions.

References

Pedersen, A.R., Petersen, S.O., Schelde, K., 2010. A comprehensive approach to soil-atmosphere trace-gas flux estimation with static chambers. European Journal of Soil Science 61, 888-902. https://doi.org/10.1111/j.1365-2389.2010.01291.x

Hutchinson, G.L., Mosier, A.R., 1981. Improved Soil Cover Method for Field Measurement of Nitrous Oxide Fluxes. Soil Science Society of America Journal 45, 311-316. https://doi.org/10.2136/sssaj1981.0361599500450

Zhao, P., Hammerle, A., Zeeman, M., Wohlfahrt, G., 2018. On the calculation of daytime CO2 fluxes measured by automated closed transparent chambers. Agricultural and Forest Meteorology 263, 267-275. https://doi.org/10.1016/j.agrformet.2018.08.022

Examples

```
data(co2_conc)
flux_fitting(co2_conc, conc, datetime, fit_type = "exp_zhao18")
flux_fitting(co2_conc, conc, datetime,  fit_type = "quadratic",
t_zero = 10, end_cut = 30)
```

`flux_flag_count` *Counts quality flags*

Description

Provides a table of how many fluxes were attributed which quality flag. This function is incorporated in flux_quality as a message, but can be used alone to extract a dataframe with the flag count.

Usage

```
flux_flag_count(
  flags_df,
  f_fluxid = f_fluxid,
  f_quality_flag = f_quality_flag,
 f_flags = c("ok", "discard", "zero", "force_discard", "start_error", "no_data",
   "force_ok", "force_zero", "force_lm", "no_slope")
)
```

Arguments

`flags_df`	dataframe of flux slopes
`f_fluxid`	column containing fluxes unique ID
`f_quality_flag`	column containing the quality flags
`f_flags`	list of flags used in the dataset (if different from default from flux_quality). If not provided, it will list only the flags that are present in the dataset (no showing 0).

Value

a dataframe with the number of fluxes for each quality flags and their proportion to the total

Author(s)

Vincent Belde

Examples

```
data(co2_conc)
slopes <- flux_fitting(co2_conc, conc, datetime, fit_type = "exp_zhao18")
slopes_flag <- flux_quality(slopes, conc)
flux_flag_count(slopes_flag)
```

`flux_gpp` *Calculates GPP*

Description

[Superseded]

See the more generic flux_diff

to calculate gross primary production (GPP) from net ecosystem (NEE) exchange and ecosystem respiration (ER) as GPP = NEE - ER. Datetime and other variables to keep will be taken from the NEE measurement. Fluxes presents in the dataset that are neither NEE nor ER (soilR, LRC or other) are not lost.

Usage

```
flux_gpp(
  fluxes_df,
  type_col,
  f_datetime,
  f_flux = f_flux,
  id_cols,
  nee_arg = "NEE",
  er_arg = "ER",
  cols_keep = "none"
)
```

Arguments

`fluxes_df`	a dataframe containing NEE and ER
`type_col`	column containing type of flux (NEE or ER)
`f_datetime`	column containing start of measurement as datetime
`f_flux`	column containing flux values
`id_cols`	columns used to identify each pair of ER and NEE
`nee_arg`	argument designating NEE fluxes in type column
`er_arg`	argument designating ER fluxes in type column
`cols_keep`	columns to keep from `fluxes_df`. Values from NEE row will be filled in GPP row. `none` (default) keeps only the columns in `id_cols`, flux, type and datetime columns; `all` keeps all the columns; can also be a vector of column names.

Value

a dataframe with $GPP = NEE - ER$ in long format with GPP, NEE, and ER as flux type, datetime, and any column specified in `cols_keep`. Values of datetime and columns in `cols_keep` for GPP row are taken from NEE measurements.

Examples

```
data(co2_fluxes)
flux_gpp(co2_fluxes, type, f_start, id_cols = "turfID",
cols_keep = c("temp_soil"))
```

`flux_lrc` *Standardizes CO2 fluxes at fixed PAR values*

Description

Calculates light response curves (LRC) for CO2 fluxes and standardizes CO2 fluxes according to the LRC

Usage

```
flux_lrc(
  fluxes_df,
  type_col,
  par_ave = par_ave,
  f_flux = f_flux,
  lrc_arg = "LRC",
  nee_arg = "NEE",
  er_arg = "ER",
  lrc_group = c(),
  par_nee = 300,
  par_er = 0
)
```

Arguments

`fluxes_df`	a dataframe containing NEE, ER and LRC measurements
`type_col`	column containing type of flux (NEE, ER, LRC)
`par_ave`	column containing the PAR value for each flux
`f_flux`	column containing flux values
`lrc_arg`	argument designating LRC fluxes in type column
`nee_arg`	argument designating NEE fluxes in type column
`er_arg`	argument designating ER fluxes in type column
`lrc_group`	character vector of columns to use to group the LRC (campaign, site, treatment), if applicable
`par_nee`	PAR value to correct the NEE fluxes to
`par_er`	PAR value to correct the ER fluxes to

Details

The light response curves are calculated with a quadratic of the form $flux(PAR) = a * PAR^2 + b * PAR + c$

The long format of the output with both uncorrected and corrected fluxes in the same flux column allows for easier gross primary production (GPP) fluxes with flux_gpp (`par_correction` will have to be added to the argument `id_cols`).

Value

the same dataframe with the additional column `par_correction` = `TRUE` for correct fluxes. Corrected fluxes are in the same `f_flux` column. Non corrected fluxes and other fluxes are kept, with NA in `par_correction`.

Examples

```
data(co2_fluxes_lrc)
flux_lrc(
fluxes_df = co2_fluxes_lrc,
type_col = type,
par_ave = PAR_ave,
f_flux = f_flux,
lrc_arg = "LRC",
nee_arg = "NEE",
er_arg = "ER",
lrc_group = c("warming"),
par_nee = 300,
par_er = 0
)
```

---

flux_match *Matching continuously measured fluxes with measurement IDs and meta data*

Description

Matching a dataframe of continuously measured gas concentration data with measurement metadata from another dataframe. Measurements are paired with their metadata based on datetime. Extra variables in both dataframes are kept in the output.

Usage

```
flux_match(
  raw_conc,
  field_record,
  f_datetime,
  start_col,
  end_col,
  measurement_length,
```

```
  fixed_length = deprecated(),
  time_diff = 0,
  startcrop = 0,
  ratio_threshold = deprecated(),
  f_conc = deprecated()
)
```

Arguments

raw_conc	dataframe of CO2 concentration measured continuously. Has to contain at least a datetime column in ymd_hms format and a gas concentration column as double.
field_record	dataframe recording which measurement happened when. Has to contain at least a column containing the start of each measurement, and any other column identifying the measurements.
f_datetime	datetime column in raw_conc (ymd_hms format)
start_col	start column in field_record (ymd_hms format)
end_col	end column in field_record (ymd_hms format), if present (see measurement_length).
measurement_length	length of the measurements (in seconds) from the start specified in the field_record. Use measurement_length if all the measurements have the same length and no end column is present in field_record.
fixed_length	**[Deprecated]** no longer required. flux_match will detect if end_col or measurement_length are provided.
time_diff	time difference (in seconds) between the two datasets. Will be added to the datetime column of the raw_conc dataset. For situations where the time was not synchronized correctly.
startcrop	**[Deprecated]** startcrop is no longer supported. Please use start_cut in flux_fitting instead.
ratio_threshold	**[Deprecated]** ratio_threshold is no longer supported. Please use ratio_threshold in flux_quality instead.
f_conc	**[Deprecated]** f_conc is no longer required

Details

If both end_col and measurement_length are provided, end_col will be ignored. Measurements either all have the same length (provide measurement_length), or the length varies and end_col has to be provided.

Value

a dataframe with concentration measurements, corresponding datetime, flux ID (f_fluxid), measurements start (f_start) and end (f_end).

Examples

```
data(co2_df_short, record_short)
flux_match(co2_df_short, record_short, datetime, start,
measurement_length = 180)
```

`flux_plot` *Plotting fluxes for visual evaluation*

Description

Plots the fluxes, fit and slope in facets with color code indicating quality flags This function takes time to run and is optional in the workflow, but it is still highly recommended to use it to visually check the measurements. Note that 'flux_plot' is specific to the fluxible package and will work best with datasets produced following a fluxible workflow.

Usage

```
flux_plot(
  slopes_df,
  f_conc = f_conc,
  f_datetime = f_datetime,
  color_discard = "#D55E00",
  color_cut = "#D55E00",
  color_ok = "#009E73",
  color_zero = "#CC79A7",
  scale_x_datetime_args = list(date_breaks = "1 min", minor_breaks = "10 sec",
    date_labels = "%e/%m \n %H:%M"),
  f_ylim_upper = 800,
  f_ylim_lower = 400,
  f_plotname = "",
  f_facetid = "f_fluxid",
  facet_wrap_args = list(ncol = 4, nrow = 3, scales = "free"),
  longpdf_args = list(ncol = 4, width = 29.7, ratio = 1),
  y_text_position = 500,
  print_plot = "FALSE",
  output = "print_only",
  ggsave_args = list()
)
```

Arguments

`slopes_df`	dataset containing slopes, with flags produced by flux_quality
`f_conc`	column with gas concentration
`f_datetime`	column with datetime of each data point
`color_discard`	color for fits with a discard quality flag
`color_cut`	color for the part of the flux that is cut

color_ok	color for fits with an ok quality flag
color_zero	color for fits with a zero quality flag
scale_x_datetime_args	list of arguments for scale_x_datetime
f_ylim_upper	y axis upper limit
f_ylim_lower	y axis lower limit
f_plotname	filename for the extracted pdf file; if empty, the name of slopes_df will be used
f_facetid	character vector of columns to use as facet IDs. Note that they will be united, and that has to result in a unique facet ID for each measurement. Default is f_fluxid
facet_wrap_args	list of arguments for facet_wrap, also used by facet_wrap_paginate in case output = "pdfpages
longpdf_args	arguments for longpdf in the form list(ncol, width (in cm), ratio)
y_text_position	position of the text box
print_plot	logical, if TRUE it prints the plot as a ggplot object but will take time depending on the size of the dataset
output	"pdfpages", the plots are saved as A4 landscape pdf pages; "ggsave", the plots can be saved with the ggsave function; "print_only" (default) prints the plot without creating a file (independently from print_plot being TRUE or FALSE); "longpdf", the plots are saved as a pdf file as long as needed (faster than "pdfpages")
ggsave_args	list of arguments for ggsave (in case output = "ggsave")

Details

output = "pdfpages" uses facet_wrap_paginate, which tends to be slow and heavy. With output = "longpdf, a long single page pdf is exported. Default width is 29.7 cm (A4 landscape) and is will be as long as it needs to be to fit all the facets. The arguments ncol and ratio in longpdf_args specify the number of columns and the ratio of the facet respectively. This method is considerably faster than pdfpages, because it bypasses facet_wrap_paginate, but is a bit less aesthetic.

Value

plots of fluxes, with raw concentration data points, fit, slope, and color code indicating quality flags and cuts. The plots are organized in facets according to flux ID, and a text box display the quality flag and diagnostics of each measurement. The plots are returned as a ggplot object if print_plot = TRUE; if print_plot = FALSE it will not return anything but will produce a file according to the output argument.

Examples

```
data(co2_conc)
slopes <- flux_fitting(co2_conc, conc, datetime, fit_type = "exp_zhao18")
slopes_flag <- flux_quality(slopes, conc)
flux_plot(slopes_flag, conc, datetime)
```

flux_quality *Assessing the quality of the fits*

Description

Indicates if the slopes provided by flux_fitting should be discarded or replaced by 0 according to quality thresholds set by user

Usage

```
flux_quality(
  slopes_df,
  f_conc = f_conc,
  f_fluxid = f_fluxid,
  f_slope = f_slope,
  f_time = f_time,
  f_start = f_start,
  f_end = f_end,
  f_fit = f_fit,
  f_cut = f_cut,
  f_pvalue = f_pvalue,
  f_rsquared = f_rsquared,
  f_slope_lm = f_slope_lm,
  f_fit_lm = f_fit_lm,
  f_b = f_b,
  force_discard = c(),
  force_ok = c(),
  force_zero = c(),
  force_lm = c(),
  force_exp = c(),
  ratio_threshold = 0.5,
  gfactor_threshold = 10,
  fit_type = c(),
  ambient_conc = 421,
  error = 100,
  pvalue_threshold = 0.3,
  rsquared_threshold = 0.7,
  rmse_threshold = 25,
  cor_threshold = 0.5,
  b_threshold = 1,
  cut_arg = "cut",
  instr_error = 5,
  kappamax = FALSE
)
```

Arguments

slopes_df	dataset containing slopes

f_conc	column containing the measured gas concentration (exponential fits)
f_fluxid	column containing unique IDs for each flux
f_slope	column containing the slope of each flux (as calculated by the flux_fitting function)
f_time	column containing the time of each measurement in seconds (exponential fits)
f_start	column with datetime of the start of the measurement (after cuts)
f_end	column with datetime of the end of the measurement (after cuts)
f_fit	column containing the modeled data (exponential fits)
f_cut	column containing the cutting information
f_pvalue	column containing the p-value of each flux (linear and quadratic fits)
f_rsquared	column containing the r squared of each flux (linear and quadratic fits)
f_slope_lm	column containing the linear slope of each flux (as calculated by the flux_fitting function)
f_fit_lm	column with the fit of the linear model. (as calculated by the flux_fitting function)
f_b	column containing the b parameter of the exponential expression (exponential fits)
force_discard	vector of fluxIDs that should be discarded by the user's decision
force_ok	vector of fluxIDs for which the user wants to keep the calculated slope despite a bad quality flag
force_zero	vector of fluxIDs that should be replaced by zero by the user's decision
force_lm	vector of fluxIDs for which the linear slope should be used by the user's decision
force_exp	vector of fluxIDs for which the exponential slope should be used by the user's decision (kappamax method)
ratio_threshold	ratio of gas concentration data points over length of measurement (in seconds) below which the measurement will be considered as not having enough data points to be considered for calculations
gfactor_threshold	threshold for the g-factor. Defines a window with its opposite outside which the flux will be flagged discard (exponential quadratic fits).
fit_type	model fitted to the data, linear, quadratic or exponential. Will be automatically filled if slopes_df was produced using flux_fitting
ambient_conc	ambient gas concentration in ppm at the site of measurement (used to detect measurement that started with a polluted setup)
error	error of the setup, defines a window outside of which the starting values indicate a polluted setup
pvalue_threshold	threshold of p-value below which the change of gas concentration over time is considered not significant (linear and quadratic fits)

`rsquared_threshold`
: threshold of r squared value below which the linear model is considered an unsatisfactory fit (linear and quadratic fits)

`rmse_threshold`
: threshold for the RMSE of each flux above which the fit is considered unsatisfactory (exponential fits)

`cor_threshold`
: threshold for the correlation coefficient of gas concentration with time below which the correlation is considered not significant (exponential fits)

`b_threshold`
: threshold for the b parameter. Defines a window with its opposite inside which the fit is considered good enough (exponential fits)

`cut_arg`
: argument defining that the data point should be cut out

`instr_error`
: error of the instrument, in the same unit as the gas concentration

`kappamax`
: logical. If `TRUE` the kappamax method will be applied.

Details

the kappamax method (Hüppi et al., 2018) selects the linear slope if $|b| > kappamax$, with $kappamax = |f_slope_lm/instr_error|$. The original kappamax method was applied to the HMR model (Pedersen et al., 2010; Hutchinson and Mosier, 1981), but here it can be applied to any exponential fit.

Value

a dataframe with added columns of quality flags (`f_quality_flag`), the slope corrected according to the quality flags (`f_slope_corr`), and any columns present in the input. It will also print a summary of the quality flags. This summary can also be exported as a dataframe using flux_flag_count

References

Pedersen, A.R., Petersen, S.O., Schelde, K., 2010. A comprehensive approach to soil-atmosphere trace-gas flux estimation with static chambers. European Journal of Soil Science 61, 888–902. https://doi.org/10.1111/j.1365-2389.2010.01291.x

Hüppi, R., Felber, R., Krauss, M., Six, J., Leifeld, J., Fuß, R., 2018. Restricting the nonlinearity parameter in soil greenhouse gas flux calculation for more reliable flux estimates. PLOS ONE 13, e0200876. https://doi.org/10.1371/journal.pone.0200876

Hutchinson, G.L., Mosier, A.R., 1981. Improved Soil Cover Method for Field Measurement of Nitrous Oxide Fluxes. Soil Science Society of America Journal 45, 311–316.

Examples

```
data(co2_conc)
slopes <- flux_fitting(co2_conc, conc, datetime, fit_type = "exp_zhao18")
flux_quality(slopes, conc)
```

flux_units *Unit conversion coefficient for fluxes*

Description

calculates the conversion coefficient for flux_calc

Usage

```
flux_units(
  flux_units,
  conc_units,
  conc_units_list = c("mmol/mol", "ppm", "ppb", "ppt"),
  amount_units = c("mol", "mmol", "umol", "nmol", "pmol"),
  surface_units = c("m2", "dm2", "cm2"),
  time_units = c("d", "h", "mn", "s")
)
```

Arguments

flux_units	desired units for the calculated fluxes. Has to be of the form amount/time/surface. Amount can be mol, mmol, umol, nmol or pmol. Time can be d (day), h (hour), mn (minute) or s (seconds). Surface can be m2, dm2 or cm2.
conc_units	units of gas concentration mmol/mol, ppm, ppb or ppt.
conc_units_list	list of possible units for gas concentration.
amount_units	list of possible units for amount.
surface_units	list of possible units for surface.
time_units	list of possible units for time.

Details

The conversion is done from umol/s/m2 and gas concentration measured in ppm.

Value

A single numerical to multiply flux values with to convert units.

Examples

```
flux_units("mol/m2/mn", "ppm")
```

raw_twogases *CO2 and CH4 concentration*

Description

CO2 and CH4 measured simultaneously

Usage

```
raw_twogases
```

Format

A tibble with 21681 rows and 4 variables

co2_conc CO2 concentration in ppm

ch4_conc CH4 concentration in ppb

datetime Datetime on the datapoint

temp_air Air temperature inside the chamber in Celsius

Examples

```
raw_twogases
```

record_liahovden *Measurements meta data at Liahovden*

Description

Measurements meta data as recorded on the field at site Liahovden

Usage

```
record_liahovden
```

Format

A tibble with 138 rows and 3 variables

turfID Unique ID of the turf in which the measurement took place.

type Type of measurement: ecosystems respiration (ER) or net ecosystem exchange (NEE).

measurement_round Round of measurement.

start Datetime at which the measurement was started.

Examples

```
record_liahovden
```

record_short *Measurements meta data*

Description

Measurements meta data as recorded on the field

Usage

```
record_short
```

Format

A tibble with 6 rows and 3 variables

turfID Unique ID of the turf in which the measurement took place.

type Type of measurement: ecosystems respiration (ER) or net ecosystem exchange (NEE).

start Datetime at which the measurement was started.

Examples

```
record_short
```

stupeflux *From raw gas concentration over time to clean fluxes*

Description

Wrapper function for the Fluxible workflow. We recommend using the step-by-step workflow for more control over the process.

Usage

```
stupeflux(
  raw_conc,
  field_record,
  f_datetime,
  start_col,
  end_col,
  f_conc,
  setup_volume,
  measurement_length,
  fit_type,
  temp_air_col,
  atm_pressure,
```

```
  plot_area,
  conc_unit,
  flux_unit,
  cols_keep = c(),
  cols_ave = c(),
  cols_sum = c(),
  cols_med = c(),
  ratio_threshold = 0.5,
  time_diff = 0,
  start_cut = 0,
  end_cut = 0,
  cz_window = 15,
  b_window = 10,
  a_window = 10,
  roll_width = 15,
  t_zero = 0,
  force_discard = c(),
  force_ok = c(),
  force_zero = c(),
  ambient_conc = 421,
  error = 100,
  pvalue_threshold = 0.3,
  rsquared_threshold = 0.7,
  rmse_threshold = 25,
  cor_threshold = 0.5,
  b_threshold = 1,
  temp_air_unit = "celsius",
  cut = TRUE,
  slope_correction = TRUE
)
```

Arguments

raw_conc	dataframe of CO2 concentration measured continuously. Has to contain at least a datetime column in ymd_hms format and a gas concentration column as double.
field_record	dataframe recording which measurement happened when. Has to contain at least a column containing the start of each measurement, and any other column identifying the measurements.
f_datetime	datetime column in raw_conc (dmy_hms format)
start_col	start column in field_record (dmy_hms format)
end_col	end column in field_record (ymd_hms format)
f_conc	concentration column in raw_conc
setup_volume	volume of the flux chamber and instrument together in L, can also be a column in case it is a variable
measurement_length	length of the measurement (in seconds) from the start specified in the field_record

fit_type	exp_zhao18, exp_tz, exp_hm, quadratic or linear. exp_zhao18 is using the exponential model $C(t) = C_m + a(t - t_z) + (C_z - C_m)\exp(-b(t - t_z))$ from Zhao et al (2018). expt_tz is a modified version which allows the user to fix t_zero: $C(t) = C_m + a * t + (C_z - C_m)\exp(-b * t)$ exp_hm is using the HM model (Pedersen et al., 2010; Hutchinson and Mosier, 1981) $C(t) = C_m + (C_z - C_m)\exp(-b * t)$
temp_air_col	column containing the air temperature used to calculate fluxes. Will be averaged with NA removed.
atm_pressure	atmospheric pressure, can be a constant (numerical) or a variable (column name)
plot_area	area of the plot in m^2, can also be a column in case it is a variable
conc_unit	unit in which the concentration of gas was measured ppm or ppb
flux_unit	unit in which the calculated flux will be mmol outputs fluxes in $mmol * m^{-2} * h^{-1}$; micromol outputs fluxes in $micromol * m^{-2} * h^{-1}$
cols_keep	columns to keep from the input to the output. Those columns need to have unique values for each flux, as distinct() is applied.
cols_ave	columns with values that should be averaged for each flux in the output. Note that NA are removed in mean calculation.
cols_sum	columns with values for which is sum is provided for each flux in the output. Note that NA are removed in sum calculation.
cols_med	columns with values for which is median is provided for each flux in the output. Note that NA are removed in median calculation.
ratio_threshold	ratio of gas concentration data points over length of measurement (in seconds) below which the measurement will be considered as not having enough data points to be considered for calculations
time_diff	time difference (in seconds) between the two datasets. Will be added to the datetime column of the raw_conc dataset. For situations where the time was not synchronized correctly.
start_cut	time to discard at the start of the measurements (in seconds)
end_cut	time to discard at the end of the measurements (in seconds)
cz_window	window used to calculate Cz, at the beginning of cut window (exp_zhao18 and exp_tz fits)
b_window	window to estimate b. It is an interval after tz where it is assumed that the model fits the data perfectly (exp_zhao18 and exp_tz fits)
a_window	window at the end of the flux to estimate a (exp_zhao18 and exp_tz fits)
roll_width	width of the rolling mean for CO2 when looking for tz, ideally same as cz_window (exp_zhao18 and exp_tz fits)
t_zero	time at which the slope should be calculated (for quadratic and exp_tz fits)
force_discard	vector of fluxIDs that should be discarded by the user's decision
force_ok	vector of fluxIDs for which the user wants to keep the calculated slope despite a bad quality flag
force_zero	vector of fluxIDs that should be replaced by zero by the user's decision

`ambient_conc` ambient gas concentration in ppm at the site of measurement (used to detect measurement that started with a polluted setup)

`error` error of the setup, defines a window outside of which the starting values indicate a polluted setup

`pvalue_threshold` threshold of p-value below which the change of gas concentration over time is considered not significant (linear and quadratic fit)

`rsquared_threshold` threshold of r squared value below which the linear model is considered an unsatisfactory fit (linear and quadratic fit)

`rmse_threshold` threshold for the RMSE of each flux above which the fit is considered unsatisfactory (`exp_zhao18` and `exp_tz` fits)

`cor_threshold` threshold for the correlation coefficient of gas concentration with time below which the correlation is considered not significant (`exp_zhao18` and `exp_tz` fits)

`b_threshold` threshold for the b parameter. Defines a window with its opposite inside which the fit is considered good enough (`exp_zhao18` and `exp_tz` fits)

`temp_air_unit` units in which air temperature was measured. Has to be either `celsius` (default), `fahrenheit` or `kelvin`.

`cut` if 'TRUE' (default), the measurements will be cut according to 'f_cut' before calculating fluxes. This has no influence on the flux itself since the slope is provided from flux_fitting, but it will influence the values of the columns in `cols_ave`.

`slope_correction` logical. If `TRUE`, the flux will be calculated with the slope corrected according to the recommendations of the quality flags.

Value

a dataframe containing flux IDs, datetime of measurements' starts, fluxes in $mmol * m^{-2} * h^{-1}$ or $micromol * m^{-2} * h^{-1}$ (`f_flux`) according to `flux_unit`, temperature average for each flux in Kelvin (`f_temp_ave`), the total volume of the setup for each measurement (`f_volume_setup`), the model used in flux_fitting, any column specified in `cols_keep`, any column specified in `cols_ave` with their value averaged over the measurement after cuts and discarding NA.

References

Pedersen, A.R., Petersen, S.O., Schelde, K., 2010. A comprehensive approach to soil-atmosphere trace-gas flux estimation with static chambers. European Journal of Soil Science 61, 888–902. https://doi.org/10.1111/j.1365-2389.2010.01291.x

Hutchinson, G.L., Mosier, A.R., 1981. Improved Soil Cover Method for Field Measurement of Nitrous Oxide Fluxes. Soil Science Society of America Journal 45, 311–316. https://doi.org/10.2136/sssaj1981.0361599500450

Zhao, P., Hammerle, A., Zeeman, M., Wohlfahrt, G., 2018. On the calculation of daytime CO2 fluxes measured by automated closed transparent chambers. Agricultural and Forest Meteorology 263, 267–275. https://doi.org/10.1016/j.agrformet.2018.08.022

Examples

```
data(co2_df_short)
data(record_short)
stupeflux(
raw_conc = co2_df_short,
field_record = record_short,
f_datetime = datetime,
start_col = start,
f_conc = conc,
measurement_length = 180,
fit_type = "exp_zhao18",
temp_air_col = temp_air,
conc_unit = "ppm",
flux_unit = "mmol",
setup_volume = 24.575,
atm_pressure = 1,
plot_area = 0.0625
)
```

---

twogases_record *Two gases field record*

Description

Two gases field record

Usage

```
twogases_record
```

Format

A tibble with 12 rows and 1 variable

start Start datetime of each flux measurement

Examples

```
twogases_record
```

wet_conc *CO2 and H2O concentration*

Description

CO2 and H2O concentration measurements

Usage

```
wet_conc
```

Format

A tibble with 18 rows and 4 variables

Time Time in format hh:mm:ss

Date Date in format yyyy-mm-dd

co2 CO2 concentration before wet air correction

h20 H2O concentration before wet air correction

Examples

```
wet_conc
```

Index