

Package: respirometr (via r-universe)

October 14, 2026

Type Package

Title An R toolbox for analysing respirometry data

Version 0.2.1

Description An R toolbox for analysing respirometry data.

License MIT + file LICENSE

URL <http://www.roald-arboel.com/respirometr/>,
<https://github.com/roaldarbol/respirometr/>

BugReports <https://github.com/roaldarbol/respirometr/issues>

Encoding UTF-8

LazyData true

Imports cli, dplyr, hms, janitor, lifecycle, readr, rlang

Suggests knitr, rmarkdown, testthat (>= 3.0.0), here, readxl, stringr,
tinyplo, tinytable, ggplot2, patchwork, performance, see,
markdown, tidyr, cfr

Config/testthat/edition 3

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

VignetteBuilder knitr

Repository <https://roaldarbol.r-universe.dev>

Date/Publication 2025-01-07 16:29:51 UTC

RemoteUrl <https://github.com/roaldarbol/respirometr>

RemoteRef HEAD

RemoteSha ccfa5e4a75eafeb871b89f35ebf2eca9b911669e

Contents

convert_unit_co2	2
normalise_drift_lm	4
normalise_drift_means	5
plot_timeseries	6
read_licor	7
read_licor_li7000	8
read_licor_li850	9

Index	10
-------	----

convert_unit_co2	<i>Convert CO2 measurements from gas analyzer data</i>
------------------	--

Description

Converts CO2 measurements from gas analyzer data into standardized units, with options for temperature correction. It can process both single measurements and vectors of measurements.

The function implements standard gas analysis conversions and can optionally apply temperature corrections using a $Q_{10} = 2$ approach (metabolic rate doubles for every 10°C increase).

Usage

```
convert_unit_co2(  
  co2_value,  
  flow_rate,  
  unit_flowrate = "mL/min",  
  unit_input_co2 = "umol/mol",  
  unit_output_time = "h",  
  unit_output_vol = "mL",  
  from_temperature = NULL,  
  to_temperature = NULL  
)
```

Arguments

- co2_value** Numeric vector. The differential CO2 value(s) from gas analyzer (default in mol/mol or ppm)
- flow_rate** Numeric. The flow rate of gas though the system.
- unit_flowrate** Character. Flow rate unit, either "mL/min" or "mL/h". Default: "mL/min"
- unit_input_co2** Character. Input CO2 unit, either "uM/M" or "umol/mol" (both equivalent to ppm). Default: "umol/mol"

unit_output_time
Character. Desired time unit for the output rate, one of "s", "min", or "h". Default: "min"

unit_output_vol
Character. Desired volume unit for the output, either "mL" or "uL". Default: "mL"

from_temperature
Numeric. Source temperature in Celsius. Only required if metabolic temperature correction is desired (must be provided with to_temperature).

to_temperature
Numeric. Optional target temperature in Celsius. Must be provided together with from_temperature if temperature correction is desired.

Details

The function performs the following conversions:

1. Converts CO2 measurements from ppm (mol/mol) to fractions
2. Optionally applies temperature correction using $Q_{10} = 2$
3. Converts to desired output units (L conversion is direct volume conversion)

Temperature correction uses the formula: $\text{correction_factor} = 10^{((\text{to_temperature} - \text{from_temperature}) * (\log_{10}(2)/10))}$

Value

Numeric vector of converted CO2 values in the specified output units

References

Temperature correction method based on standard $Q_{10} = 2$ approach, where metabolic rate doubles for every 10°C increase in temperature.

Examples

```
# Basic usage
convert_unit_co2(
  co2_value = 100,
  flow_rate = 1300,
  unit_output_time = "h",
  unit_output_vol = "uL"
)

# With temperature correction
convert_unit_co2(
  co2_value = 100,
  flow_rate = 200,
  from_temperature = 17, # Experiment temperature
  to_temperature = 25,   # Standard temperature
  unit_output_vol = "uL",
```

```

    unit_output_time = "min"
  )

  # Processing a sequence of measurements
  co2_values <- c(95, 98, 100, 103, 99)
  convert_unit_co2(
    co2_value = co2_values,
    flow_rate = 200,
    unit_output_vol = "mL",
    unit_output_time = "min"
  )

```

normalise_drift_lm	<i>Normalize Measurement Data Using Linear Regression</i>
--------------------	---

Description

This function normalizes measurement data by subtracting baseline values predicted from a linear regression of pre and post data.

Usage

```

normalise_drift_lm(
  data,
  data_pre,
  data_post,
  colname_time = "time",
  colname_measure = "co2d_um_m"
)

```

Arguments

<code>data</code>	A data frame containing the main dataset to be normalized
<code>data_pre</code>	A data frame containing pre-experiment data used for baseline calculation
<code>data_post</code>	A data frame containing post-experiment data used for baseline calculation
<code>colname_time</code>	Character string specifying the column name for time measurements. Default is "time".
<code>colname_measure</code>	Character string specifying the column name for the measurement. Default is "co2d_um_m".

Details

The function performs the following steps:

1. Combines pre and post data
2. Fits a linear regression model using time as the predictor
3. Predicts baseline values for the main dataset
4. Subtracts these baseline values from the original measurements

Value

A modified version of the input **data** with baseline-adjusted measurements

Examples

```
## Not run:
# Assuming you have pre, post, and main datasets
normalized_data <- normalise_drift_lm(
  data = main_data,
  data_pre = pre_experiment_data,
  data_post = post_experiment_data
)

## End(Not run)
```

`normalise_drift_means`

Normalize Data for Linear Drift Between Pre and Post Measurements

Description

This function normalizes a data vector by subtracting a linear interpolation between the means of pre and post measurements. This is useful for correcting measurement drift in experimental data.

Usage

```
normalise_drift_means(data, pre, post)
```

Arguments

data	Numeric vector containing the measurements to be normalized
pre	Numeric vector or single value representing pre-measurement(s)
post	Numeric vector or single value representing post-measurement(s)

Details

The function performs the following steps:

1. Calculates mean of pre-measurements if multiple values provided
2. Calculates mean of post-measurements if multiple values provided
3. Creates a linear interpolation between pre and post means
4. Subtracts interpolated values from the data to correct for drift

Value

A numeric vector of the same length as **data** containing the normalized values

Examples

```
# Single pre/post values
data <- c(15, 16, 17, 18, 19)
normalise_drift_means(data, pre = 10, post = 20)

# Multiple pre/post values
normalise_drift_means(
  data = c(15, 16, 17, 18, 19),
  pre = c(10, 11, 12),
  post = c(19, 20, 21)
)
```

plot_timeseries	<i>Plot Time Series of Values</i>
-----------------	-----------------------------------

Description

Creates a visualization of time series data. Useful for analyzing patterns in continuous measurements such as physiological data or other single-trace recordings.

Usage

```
plot_timeseries(data, value_col, time_col = "time", y_max = NULL)
```

Arguments

data	A data frame containing time series data with the following columns: • Time column (specified by time_col, defaults to "time") • Column specified by value_col containing the values to plot
value_col	Character string specifying which column to plot
time_col	Character string specifying the time column (default: "time")
y_max	Optional numeric value specifying the maximum value for the y-axis. If NULL (default), the y-axis limit is automatically determined from the data.

Value

A ggplot object wrapped in patchwork.

Examples

```
## Not run:
# Plot heart rate data
plot_timeseries(physio_data, value_col = "heart_rate")

# Plot temperature with custom time column
plot_timeseries(physio_data,
                value_col = "temperature",
                time_col = "timestamp")

## End(Not run)
```

read_licor*Read Licor data*

Description

Read respirometry data from a variety of Licor devices.

Usage

```
read_licor(filepath, model)
```

Arguments

filepath	Path to file
model	LiCor model

Value

Tidy LiCor data

read_licor_li7000	<i>Read and Process LI-7000 Data Files</i>
-------------------	--

Description

Reads data from a LI-7000 CO₂/H₂O analyzer file, cleans column names, and ensures proper time column handling. Handles cases where headers are repeated mid-file.

Usage

```
read_licor_li7000(filepath)
```

Arguments

filepath	Character string specifying the path to the LI-7000 data file. File should be tab-separated with two header rows.
----------	---

Details

The function performs the following operations:

1. Reads the entire file as lines
2. Identifies and removes any mid-file headers
3. Cleans column names using `janitor::make_clean_names()`
4. Ensures proper time column naming

Value

A tibble containing the processed LI-7000 data with:

- All original columns with cleaned names (lowercase, no spaces)
- Standardized time column
- Data combined across any mid-file header breaks

Examples

```
## Not run:  
# Read a LI-7000 data file  
data <- read_licor_li7000("path/to/li7000_data.txt")  
  
## End(Not run)
```

`read_licor_li850`*Read and Process LI-850 Data Files*

Description

Reads data from a LI-850 CO₂/H₂O analyzer file, cleans column names, removes columns with NA values, and adds an elapsed time column. Handles cases where headers are repeated mid-file.

Usage

```
read_licor_li850(filepath)
```

Arguments

<code>filepath</code>	Character string specifying the path to the LI-850 data file. File should be tab-separated with a one-line header.
-----------------------	--

Details

The function performs the following operations:

1. Reads the entire file and handles any mid-file headers
2. Cleans column names using `janitor::make_clean_names()`
3. Removes any columns containing NA values
4. Converts system time to HMS format
5. Adds an elapsed time column in seconds

Value

A tibble containing the processed LI-850 data with:

- All original columns except those containing NA values
- Cleaned column names (lowercase, no spaces)
- `system_time_h_m_s` converted to HMS format
- New `time` column with seconds elapsed from start
- Data combined across any mid-file header breaks

Examples

```
## Not run:  
# Read a LI-850 data file  
data <- read_licor_li850("path/to/li850_data.txt")  
  
## End(Not run)
```

Index

`convert_unit_co2`, [2](#)

`normalise_drift_lm`, [4](#)
`normalise_drift_means`, [5](#)
`normalize_drift_lm`
 (*normalise_drift_lm*), [4](#)
`normalize_drift_means`
 (*normalise_drift_means*), [5](#)

`plot_timeseries`, [6](#)

`read_licor`, [7](#)
`read_licor_li7000`, [8](#)
`read_licor_li850`, [9](#)