Package 'aelab'

May 7, 2026

Type Package

Title Data Processing for Aquatic Ecology

Version 1.1.3

Maintainer Zhao-Jun Yong <nuannuan0425@gmail.com>

Description Facilitate the analysis of data related to aquatic ecology, specifically the establishment of carbon budget.
Currently, the package allows the below analysis.
(i) the calculation of greenhouse gas flux based on data obtained from trace gas analyzer using the method described in Lin et al. (2024).
(ii) the calculation of Dissolved Oxygen (DO) metabolism based on data obtained from dissolved oxygen data logger using the method described in Staehr et al. (2010).
Yong et al. (2024) <doi:10.5194/bg-21-5247-2024>.
Staehr et al. (2010) <doi:10.4319/lom.2010.8.0628>.

Imports tibble, lubridate, stats, dplyr, openxlsx, readxl, ggplot2, readr, tidyr, stringr, purrr, rlang, grDevices, multcompView, FSA, rcompanion, rnaturalearth, sf, ggspatial

License GPL (>= 3)

Encoding UTF-8

RoxygenNote 7.2.3

Suggests knitr, rmarkdown, testthat (>= 3.0.0), rnaturalearthdata

Config/testthat/edition 3

Depends R (>= 4.1.0)

VignetteBuilder knitr

LazyData true

NeedsCompilation no

Author Zhao-Jun Yong [cre, aut]

Repository CRAN

Date/Publication 2026-05-07 12:10:44 UTC

Contents

`aelab_palettes` *aelab_palettes*

Description

Retrieve a named aelab colour palette as a character vector.

Usage

```
aelab_palettes(name, n, type = c("discrete", "continuous"))
```

Arguments

`name`	Name of the palette (string).
`n`	Number of colours to return. Defaults to the full palette length.
`type`	`"discrete"` (default) or `"continuous"`. For `"continuous"`, colours are interpolated to length `n`.

Details

Available palette names: `"rainbow"`, `"two"`, `"control"`, `"control2"`, `"control3"`, `"period"`, `"ghg"`.

Value

A character vector of hex colour codes with class `"palette"`.

Examples

```
aelab_palettes("rainbow", 5)
aelab_palettes("ghg", type = "continuous", n = 20)
```

`aov_test` *aov_test*

Description

Perform one-way ANOVA followed by Tukey HSD post-hoc test with compact letter display.

Usage

```
aov_test(df, variable_name, group)
```

Arguments

df	A data frame.
variable_name	Name of the response variable column (string).
group	Name of the grouping column (string).

Value

A named list with elements anova_summary, tukey_results, and compact_letters.

Examples

```
df <- data.frame(
  grp = rep(c("A","B","C"), each = 5),
  val = c(1,2,1,2,1, 3,4,3,4,3, 5,6,5,6,5)
)
aov_test(df, "val", "grp")
```

---

calculate_do *calculate_do*

Description

Calculate the Net Ecosystem Production, Gross Primary Production and Ecosystem respiration based on the change in dissolved oxygen concentration.

Usage

```
calculate_do(df)
```

Arguments

df	Merged dataframe produced by process_hobo(), process_weather() and process_info() functions.

Value

A dataframe.

Examples

```
data(hobo)
calculate_do(hobo)
```

calculate_ghg_flux *calculate_ghg_flux*

Description

Calculate the greenhouse gas (GHG) flux based on input parameters from a data frame.

Usage

```
calculate_ghg_flux(
  data,
  slope = "slope",
  area = "area",
  volume = "volume",
  temp = "temp"
)
```

Arguments

data	A data frame containing relevant data with columns for slope, area, volume, and temperature.
slope	Name of the column in 'data' that contains the slope values of the GHG concentration change (in ppm/s).
area	Name of the column in 'data' that contains the values of the area of the chamber (in square meter).
volume	Name of the column in 'data' that contains values of the volume of the chamber (in litre).
temp	Name of the column in 'data' that contains values of the temperature of the gas (in Celsius).

Value

A list containing the calculated flux and its unit.

Examples

```
data <- data.frame(
  slope = c(1.2, 1.5, 1.1),
  area = c(100, 150, 120),
  volume = c(10, 15, 12),
  temp = c(25, 30, 22)
)
results <- calculate_ghg_flux(data)
print(results)
```

calculate_MDF *calculate_MDF*

Description

Calculate the Minimum Detectable Flux (MDF) for a static chamber GHG measurement system.

Usage

```
calculate_MDF(
  precision_ppm,
  closure_time_s,
  data_point_n,
  chamber_volume_m3,
  temperature_C,
  chamber_area_m2,
  pressure_pa = 101325,
  ideal_constant = 8.314,
  ghg = "co2"
)
```

Arguments

precision_ppm	Precision of the gas analyser (ppm).
closure_time_s	Closure time of the measurement (seconds).
data_point_n	Number of data points recorded during the closure period.
chamber_volume_m3	Internal volume of the chamber (m^3).
temperature_C	Air temperature at the measurement location (°C).
chamber_area_m2	Base area of the chamber (m^2).
pressure_pa	Atmospheric pressure (Pa). Default 101325.
ideal_constant	Ideal gas constant (J mol^{-1} K^{-1}). Default 8.314.
ghg	Greenhouse gas type: "co2", "ch4", or "n2o". Default "co2".

Value

A named list with MDF (numeric, μg m^{-2} h^{-1}) and unit (string).

Examples

```
calculate_MDF(
  precision_ppm     = 1,
  closure_time_s    = 300,
  data_point_n      = 300,
```

```
  chamber_volume_m3 = 0.0064,
  temperature_C     = 25,
  chamber_area_m2   = 0.07
)
```

`calculate_regression` *calculate_regression*

Description

Calculate the slope of greenhouse gas (GHG) concentration change over time using simple linear regression.

Usage

```
calculate_regression(
  data,
  reference_df,
  ghg,
  reference_time,
  site,
  analyzer_code,
  duration_minutes = 7,
  num_rows = 300
)
```

Arguments

`data`	Data from the GHG analyzer that has been processed and time-converted.
`reference_df`	A data frame containing measurement reference times, site names, and an `analyzer` column.
`ghg`	Column name in `data` containing GHG concentration values (e.g., `"CH4"`, `"N2O"`).
`reference_time`	Column name in `reference_df` containing the measurement start times (POSIXct).
`site`	Column name in `reference_df` containing site identifiers.
`analyzer_code`	String pattern used to filter results by the `analyzer` column of `reference_df`.
`duration_minutes`	The duration of the measurement, default to 7.
`num_rows`	The number of rows used to perform the regression, default to 300.

Value

A tibble containing the time range, slope, R2, site, and analyzer from the simple linear regression.

Examples

```
data(n2o)
n2o_converted <- convert_time(n2o)
ref <- data.frame(
  date_time = n2o_converted$real_datetime[1],
  site = "S1",
  analyzer = "1074"
)
calculate_regression(n2o_converted, ref, "N2O",
                     reference_time = "date_time", site = "site",
                     analyzer_code = "1074")
```

`calculate_total_co2e` *calculate_total_co2e*

Description

Convert individual GHG fluxes (mg m^{-2} h^{-1}) to a total CO_2-equivalent flux (g m^{-2} d^{-1}) using IPCC AR6 100-year GWPs (CO_2 = 1, CH_4 = 27, N_2O = 273).

Usage

```
calculate_total_co2e(co2 = 0, ch4 = 0, n2o = 0)
```

Arguments

`co2`	CO_2 flux in mg m^{-2} h^{-1}. Default 0.
`ch4`	CH_4 flux in mg m^{-2} h^{-1}. Default 0.
`n2o`	N_2O flux in mg m^{-2} h^{-1}. Default 0.

Value

Total CO_2e flux as a numeric scalar (g m^{-2} d^{-1}), printed with a diagnostic message.

Examples

```
calculate_total_co2e(co2 = 4.02, ch4 = 0.001, n2o = 0.003)
```

calc_chla_trichromatic *calc_chla_trichromatic*

Description

Calculate chlorophyll-a concentration from trichromatic spectrophotometric absorbance readings using the Jeffrey & Humphrey (1975) equations.

Usage

```
calc_chla_trichromatic(wl_630, wl_647, wl_664, wl_750)
```

Arguments

wl_630	Absorbance at 630 nm.
wl_647	Absorbance at 647 nm.
wl_664	Absorbance at 664 nm.
wl_750	Absorbance at 750 nm (turbidity blank).

Details

Absorbance values should be measured in a 1 cm path-length cuvette. The 750 nm reading is used as a turbidity blank correction. Formula: $11.85 \times E_{664} - 1.54 \times E_{647} - 0.08 \times E_{630}$ where $E_\lambda = A_\lambda - A_{750}$.

Value

Chlorophyll-a concentration in µg L^{-1} (assuming a 1 cm path length and standard extraction volume).

Examples

```
calc_chla_trichromatic(wl_630 = 0.05, wl_647 = 0.08, wl_664 = 0.20, wl_750 = 0.01)
```

combine_hobo *combine_hobo*

Description

Tidy multiple data retrieved from HOBO U26 Dissolved Oxygen Data Logger.

Usage

```
combine_hobo(file_path, file_prefix = "no.")
```

Arguments

file_path	Directory of the folder containing the files.
file_prefix	The prefix before the code for the data logger, defaults to "no."

Value

A dataframe.

Examples

```
hobo_data_path <- system.file("extdata", package = "aelab")
df <- combine_hobo(hobo_data_path, file_prefix = "ex_ho")
```

---

combine_weather *combine_weather*

Description

Tidy multiple daily weather data downloaded from weather station in Taiwan.

Usage

```
combine_weather(file_path, start_date, end_date, zone)
```

Arguments

file_path	Directory of folder containing the files (including the character in the file name that precedes the date).
start_date	Date of the daily weather data in yyyy-mm-dd format.
end_date	Date of the daily weather data in yyyy-mm-dd format.
zone	Code for the region of the weather station.

Value

A dataframe.

Examples

```
weather_data_path <- system.file("extdata", package = "aelab")
modified_data_path <- paste0(weather_data_path, "/ex_")
df <- combine_weather(modified_data_path,
start_date = "2024-01-01",
end_date = "2024-01-02", "site_A")
```

`combine_weather_month` *combine_weather_month*

Description

Batch-import monthly weather CSV files from a Taiwan Central Weather Administration station for a consecutive range of months.

Usage

```
combine_weather_month(file_path, start_month, end_month, year = 2024, zone)
```

Arguments

`file_path`	Path prefix (directory + filename prefix before the date portion, e.g. `"data/weather/"`).
`start_month`	First month to import (1–9; two-digit months not yet supported).
`end_month`	Last month to import.
`year`	Four-digit year. Default 2024.
`zone`	Character label for the weather station / region.

Details

File names are expected to follow the pattern `<file_path><year>-0<month>.csv` (e.g. `2024-01.csv`).

Value

A combined data frame produced by `process_weather_month`.

Examples

```
## Not run:
df <- combine_weather_month("data/weather/", start_month = 1,
                            end_month = 6, year = 2024, zone = "site_A")

## End(Not run)
```

convert_ghg_unit *convert_ghg_unit*

Description

Convert a greenhouse gas (GHG) flux value (or a character string containing one or more numeric values, e.g. "0.002 +/- 0.003") to micrograms per square meter per hour.

Usage

```
convert_ghg_unit(
  input,
  ghg,
  mass = "µg",
  area = "m2",
  time = "hr",
  digits = 2,
  ratio = FALSE
)
```

Arguments

input	A single numeric value or a character string containing one or more numbers.
ghg	The molecular formula of the greenhouse gas: "co2", "ch4", or "n2o".
mass	Mass unit of the input flux. One of "mmol", "mg", "g", "ug" (micrograms), "nmol", "Mg", "umol" (micromoles), "mol". Default "ug".
area	Area unit of the input flux. One of "ha", "m2". Default "m2".
time	Time unit of the input flux. One of "yr", "day", "hr", "sec", "min". Default "hr".
digits	Number of decimal places to round to. Default 2.
ratio	Logical. If TRUE, apply an elemental-ratio correction (C-basis for CH4, N-basis for N2O). Default FALSE.

Details

Numeric values embedded in a string (e.g. mean +/- SD notation) are each converted individually and the surrounding text is preserved. Commas are treated as decimal separators.

Value

A named list with value (converted string) and unit, or "EMPTY" for missing/non-numeric input.

Examples

```
convert_ghg_unit(97, ghg = "ch4", mass = "mg", area = "m2", time = "hr")
```

convert_time *convert_time*

Description

Convert the time of the LI-COR Trace Gas Analyzer to match the time in real life.

Usage

```
convert_time(data, day = 0, hr = 0, min = 0, sec = 0)
```

Arguments

data	Data from the LI-COR Trace Gas Analyzer that had been processed by tidy_licor().
day	Day(s) to add or subtract.
hr	Hour(s) to add or subtract.
min	Minute(s) to add or subtract.
sec	Second(s) to add or subtract.

Value

The input data with a new column in POSIXct format converted based on the input value.

Examples

```
data(n2o)
converted_n2o <- convert_time(n2o, min = -10, sec = 5)
```

descriptive_statistic *descriptive_statistic*

Description

Compute grouped mean ± SD and min–max summary statistics for one or more numeric variables.

Usage

```
descriptive_statistic(data, vars, groups, digits = 2)
```

Arguments

data	A data frame.
vars	<[‘tidy-select‘][dplyr::dplyr_tidy_select]> Columns to summarise.
groups	<[‘tidy-select‘][dplyr::dplyr_tidy_select]> Grouping columns.
digits	Number of decimal places to round to. Default is 2.

Value

A tibble with one row per group and two summary columns per variable (‘<var>_mean_sd‘ and ‘<var>_min_max‘).

Examples

```
df <- data.frame(group = c("A","A","B","B"), value = c(1.1, 2.3, 3.5, 4.7))
descriptive_statistic(df, vars = value, groups = group)
```

`df_trans` *df_trans*

Description

Apply a reverse square-root or reverse log transformation to a numeric column and append the result as a new column.

Usage

```
df_trans(df, variable_name, transformation)
```

Arguments

`df`	A data frame.
`variable_name`	Name of the column to transform (string).
`transformation`	Transformation type: `"sqrt"` or `"log"`.

Value

The input data frame with an additional column named `<variable_name>_sqrt` or `<variable_name>_log`.

Examples

```
df <- data.frame(val = c(1, 4, 9, 16))
df_trans(df, "val", "sqrt")
```

find_outlier *find_outlier*

Description

Identify outliers in a numeric column using the IQR method (values outside 1.5 × IQR from Q1/Q3).

Usage

```
find_outlier(df, var, other_var = NULL)
```

Arguments

df	A data frame.
var	Name of the column to check for outliers (string).
other_var	Character vector of additional column names to return alongside the outlier values, or NULL.

Value

A tibble with columns row_index, outlier_value, and any requested other_var columns.

Examples

```
df <- data.frame(val = c(1, 2, 2, 3, 100), id = 1:5)
find_outlier(df, "val", "id")
```

hobo *Processed data from Onset HOBO Dissolved Oxygen Data Logger. A dataset containing 336 dissolved oxygen concentrations changed over time.*

Description

Processed data from Onset HOBO Dissolved Oxygen Data Logger. A dataset containing 336 dissolved oxygen concentrations changed over time.

Format

A data.frame with 336 rows and 13 variables:

- date_time: Date and time in POSIXct format.
- pressure_hpa: Atmospheric pressure (hpa).
- wind_ms: Wind speed (m/s).
- do: Dissolved oxygen concentrations (mg/L)
- temp: Water temperature (Celsius)
- depth: Water depth (m).
- salinity: Salinity (ppt).
- start_date_time: Start date and time of the deployment.
- end_date_time: End date and time of the deployment.
- sunrise: Sunrise time during that day.
- sunset: Sunset time during that day.
- no_hobo: Name for the data logger .
- site: Name for the site.

Source

own data.

`ks_test` *ks_test*

Description

Perform Kruskal-Wallis test followed by Dunn post-hoc test (Bonferroni correction) with compact letter display.

Usage

```
ks_test(df, variable_name, group)
```

Arguments

`df`	A data frame.
`variable_name`	Name of the response variable column (string).
`group`	Name of the grouping column (string).

Value

A named list with elements `ks_results`, `dunn_results`, `mean_summary`, and `compact_letters`.

Examples

```
df <- data.frame(
  grp = rep(c("A","B","C"), each = 5),
  val = c(1,2,1,2,1, 3,4,3,4,3, 5,6,5,6,5)
)
ks_test(df, "val", "grp")
```

`n2o` *Processed data from N2O LI-COR Trace Gas Analyzer. A dataset containing 567 N2O concentrations changed over time.*

Description

Processed data from N2O LI-COR Trace Gas Analyzer. A dataset containing 567 N2O concentrations changed over time.

Format

A data.frame with 567 rows and 4 variables:

- DATE: Date in character format.
- TIME: Time in character format.
- N2O: Concentrations of nitrous oxide (N2O), in ppb.
- date_time: Date and time in POSIXct format.

Source

own data.

`normality_test_aov` *normality_test_aov*

Description

Test normality of ANOVA model residuals using Shapiro-Wilk on raw, square-root, and log10 transforms (one-way or two-way).

Usage

```
normality_test_aov(df, variable_name, group_1, group_2 = NULL)
```

Arguments

df	A data frame.
variable_name	Name of the response variable column (string).
group_1	Name of the first grouping column (string).
group_2	Name of the second grouping column (string), or NULL for a one-way model.

Value

A tibble with Shapiro-Wilk p-values for each transformation.

Examples

```
df <- data.frame(
  grp = c("A","A","B","B"),
  val = c(1.1, 1.4, 3.2, 3.8)
)
normality_test_aov(df, "val", "grp")
```

---

normality_test_t *normality_test_t*

Description

Test normality of a variable within two groups using Shapiro-Wilk on raw, square-root, and log10 transforms (for t-test context).

Usage

```
normality_test_t(df, variable_name, group, group_1, group_2)
```

Arguments

df	A data frame.
variable_name	Name of the numeric variable column (string).
group	<[‘data-masking‘][dplyr::dplyr_data_masking]> The grouping column.
group_1	Value identifying the first group.
group_2	Value identifying the second group.

Value

A tibble with Shapiro-Wilk p-values for each group × transformation combination.

Examples

```
df <- data.frame(
  grp = c("A","A","A","B","B","B"),
  val = c(1.1, 2.0, 1.5, 4.2, 3.8, 4.5)
)
normality_test_t(df, "val", grp, "A", "B")
```

---

plot_bar *plot_bar*

Description

Create a bar plot using the aelab theme.

Usage

```
plot_bar(
  df,
  x,
  y,
  z = NULL,
  base_size = 25,
  line_width = 1,
  text_color = "black",
  facet = FALSE,
  facet_x = NULL,
  facet_y = NULL,
  style = "bw",
  position = "dodge",
  stat = "identity"
)
```

Arguments

df	A data frame.
x	<[‘data-masking‘][ggplot2::aes]> Column mapped to the x-axis.
y	<[‘data-masking‘][ggplot2::aes]> Column mapped to the y-axis.
z	<[‘data-masking‘][ggplot2::aes]> Optional column mapped to fill colour.
base_size	Base font size. Default 25.
line_width	Bar outline width. Default 1.
text_color	Text colour. Default "black".
facet	Logical; add facet grid? Default FALSE.
facet_x	Column name (string) for the horizontal facet dimension.
facet_y	Column name (string) for the vertical facet dimension.

style	Theme style. Default "bw".
position	Bar position: "dodge" or "stack". Default "dodge".
stat	Stat type: "identity" or "count". Default "identity".

Value

A ggplot object.

Examples

```
## Not run:
df <- data.frame(x = c("A","B","A","B"), g = c("X","X","Y","Y"), y = c(1,2,3,4))
plot_bar(df, x, y, g)

## End(Not run)
```

---

plot_box *plot_box*

Description

Create a box plot with mean overlay using the aelab theme.

Usage

```
plot_box(
  df,
  x,
  y,
  z = NULL,
  base_size = 25,
  line_width = 0.5,
  outlier_size = 1.5,
  text_color = "black",
  facet = FALSE,
  facet_x = NULL,
  facet_y = NULL,
  style = "bw"
)
```

Arguments

df	A data frame.
x	<[‘data-masking‘][ggplot2::aes]> Column mapped to the x-axis.
y	<[‘data-masking‘][ggplot2::aes]> Column mapped to the y-axis.
z	<[‘data-masking‘][ggplot2::aes]> Optional column mapped to fill colour.

base_size	Base font size. Default 25.
line_width	Box outline width. Default 0.5.
outlier_size	Outlier point size. Default 1.5.
text_color	Text colour. Default "black".
facet	Logical; add facet grid? Default FALSE.
facet_x	Column name (string) for the horizontal facet dimension.
facet_y	Column name (string) for the vertical facet dimension.
style	Theme style. Default "bw".

Value

A ggplot object.

Examples

```
## Not run:
df <- data.frame(x = rep(c("A","B"), each = 5), y = c(1:5, 3:7))
plot_box(df, x, y)

## End(Not run)
```

---

plot_hobo *plot_hobo*

Description

Plot the dissolved oxygen concentration over time series grouped by different data loggers to observe the variations.

Usage

```
plot_hobo(df)
```

Arguments

df	Dataframe produced by process_hobo() function.

Value

A plot generated by ggplot2.

Examples

```
data(hobo)
plot_hobo(hobo)
```

plot_line *plot_line*

Description

Create a line plot using the aelab theme.

Usage

```
plot_line(
  df,
  x,
  y,
  z = NULL,
  base_size = 25,
  line_width = 3,
  text_color = "black",
  facet = FALSE,
  facet_x = NULL,
  facet_y = NULL,
  style = "bw"
)
```

Arguments

df	A data frame.
x	<[‘data-masking‘][ggplot2::aes]> Column mapped to the x-axis.
y	<[‘data-masking‘][ggplot2::aes]> Column mapped to the y-axis.
z	<[‘data-masking‘][ggplot2::aes]> Optional column mapped to colour and group.
base_size	Base font size. Default 25.
line_width	Line width. Default 3.
text_color	Text colour. Default "black".
facet	Logical; add facet grid? Default FALSE.
facet_x	Column name (string) for the horizontal facet dimension.
facet_y	Column name (string) for the vertical facet dimension.
style	Theme style. Default "bw".

Value

A ggplot object.

Examples

```
## Not run:
df <- data.frame(x = 1:6, y = c(1,3,2,5,4,6), g = rep(c("A","B"), 3))
plot_line(df, x, y, g)

## End(Not run)
```

`plot_map_taiwan` *plot_map_taiwan*

Description

Plot sampling sites on a map of Taiwan with a north arrow and scale bar.

Usage

```
plot_map_taiwan(
  long,
  lat,
  names,
  color = "darkgrey",
  textsize = 5,
  basesize = 16,
  shape_type = 22
)
```

Arguments

`long`	Numeric vector of longitudes.
`lat`	Numeric vector of latitudes.
`names`	Character vector of site labels (same length as `long`).
`color`	Fill colour for site markers. Default `"darkgrey"`.
`textsize`	Size for annotation and point labels. Default 5.
`basesize`	Base font size for the map theme. Default 16.
`shape_type`	ggplot2 point shape number. Default 22 (filled square).

Value

A `ggplot` object (also printed to the active device).

Examples

```
## Not run:
plot_map_taiwan(
  long = c(120.2, 121.5),
  lat  = c(22.9, 24.1),
  names = c("Site A", "Site B")
)

## End(Not run)
```

`plot_point` *plot_point*

Description

Create a scatter plot using the aelab theme.

Usage

```
plot_point(
  df,
  x,
  y,
  z = NULL,
  base_size = 25,
  point_size = 3,
  stroke_size = 1,
  text_color = "black",
  facet = FALSE,
  facet_x = NULL,
  facet_y = NULL,
  style = "bw"
)
```

Arguments

`df`	A data frame.
`x`	<[‘data-masking‘][ggplot2::aes]> Column mapped to the x-axis.
`y`	<[‘data-masking‘][ggplot2::aes]> Column mapped to the y-axis.
`z`	<[‘data-masking‘][ggplot2::aes]> Optional column mapped to fill colour.
`base_size`	Base font size passed to the ggplot2 theme. Default 25.
`point_size`	Point size. Default 3.
`stroke_size`	Point stroke width. Default 1.
`text_color`	Text colour. Default `"black"`.
`facet`	Logical; add facet grid? Default `FALSE`.

facet_x	Column name (string) for the horizontal facet dimension.
facet_y	Column name (string) for the vertical facet dimension.
style	Theme style. One of "bw", "minimal", "classic", "graycolor", "light". Default "bw".

Value

A ggplot object.

Examples

```
## Not run:
df <- data.frame(x = 1:5, y = c(2,4,1,5,3), g = c("A","A","B","B","A"))
plot_point(df, x, y, g)

## End(Not run)
```

---

process_hobo *process_hobo*

Description

Tidy the data retrieved from HOBO U26 Dissolved Oxygen Data Logger.

Usage

```
process_hobo(file_path, no_hobo)
```

Arguments

file_path	Directory of file.
no_hobo	The code for the data logger.

Value

A dataframe.

Examples

```
hobo_data_path <- system.file("extdata", "ex_hobo.csv", package = "aelab")
df <- process_hobo(hobo_data_path, "code_for_logger")
```

process_info *process_info*

Description

Import and process the necessary information, including the sunrise and sunset times of the day, the date and time range of the deployment, and the code for the data logger.

Usage

```
process_info(file_path)
```

Arguments

file_path	Directory of file.

Value

A dataframe.

Examples

```
info_data_path <- system.file("extdata", "info.xlsx", package = "aelab")
df <- process_info(info_data_path)
```

process_weather *convert_time*

Description

Tidy the daily weather data downloaded from weather station in Taiwan.

Usage

```
process_weather(file_path, date, zone)
```

Arguments

file_path	Directory of file.
date	Date of the daily weather data in yyyy-mm-dd format.
zone	Code for the region of the weather station.

Value

A dataframe.

Examples

```
weather_data_path <- system.file("extdata", "ex_weather.csv", package = "aelab")
df <- process_weather(weather_data_path, "2024-01-01", "site_A")
```

`process_weather_month` *process_weather_month*

Description

Import and tidy a monthly weather CSV file downloaded from a Taiwan Central Weather Administration station. Column selection is done via regex so minor header changes are handled gracefully.

Usage

```
process_weather_month(file_path, month, year = 2024, zone)
```

Arguments

`file_path`	Path to the monthly CSV file.
`month`	Month number (1–12) covered by the file.
`year`	Four-digit year. Default 2024.
`zone`	Character label for the weather station / region.

Value

A data frame with columns `day`, `pressure_hpa`, `temp`, `humidity_percent`, `wind_ms`, `rain_mm`, `daylight_hr`, `radiation`, `date`, and `zone`.

Examples

```
## Not run:
df <- process_weather_month("path/to/2024-01.csv", month = 1, year = 2024,
                            zone = "site_A")

## End(Not run)
```

scale_colour_aelab_c *scale_colour_aelab_c*

Description

Continuous ggplot2 colour scale using an aelab palette.

Usage

```
scale_colour_aelab_c(name, direction = 1)

scale_color_aelab_c(name, direction = 1)
```

Arguments

name	Palette name passed to aelab_palettes.
direction	1 (default) for normal order; -1 to reverse.

Value

A ggplot2 scale.

Examples

```
## Not run:
ggplot2::ggplot(mtcars, ggplot2::aes(wt, mpg, colour = mpg)) +
  ggplot2::geom_point() + scale_colour_aelab_c("rainbow")

## End(Not run)
```

scale_colour_aelab_d *scale_colour_aelab_d*

Description

Discrete ggplot2 colour scale using an aelab palette.

Usage

```
scale_colour_aelab_d(name, direction = 1)

scale_color_aelab_d(name, direction = 1)
```

Arguments

name	Palette name passed to aelab_palettes.
direction	1 (default) for normal order; -1 to reverse.

Value

A ggplot2 scale.

Examples

```
## Not run:
ggplot2::ggplot(mtcars, ggplot2::aes(wt, mpg, colour = factor(cyl))) +
  ggplot2::geom_point() + scale_colour_aelab_d("rainbow")

## End(Not run)
```

---

scale_fill_aelab_c *scale_fill_aelab_c*

Description

Continuous ggplot2 fill scale using an aelab palette.

Usage

```
scale_fill_aelab_c(name, direction = 1)
```

Arguments

name	Palette name passed to aelab_palettes.
direction	1 (default) for normal order; -1 to reverse.

Value

A ggplot2 scale.

Examples

```
## Not run:
ggplot2::ggplot(mtcars, ggplot2::aes(factor(cyl), fill = mpg)) +
  ggplot2::geom_col() + scale_fill_aelab_c("ghg")

## End(Not run)
```

scale_fill_aelab_d *scale_fill_aelab_d*

Description

Discrete ggplot2 fill scale using an aelab palette.

Usage

```
scale_fill_aelab_d(name, direction = 1)
```

Arguments

name	Palette name passed to aelab_palettes.
direction	1 (default) for normal order; -1 to reverse.

Value

A ggplot2 scale.

Examples

```
## Not run:
ggplot2::ggplot(mtcars, ggplot2::aes(factor(cyl), fill = factor(cyl))) +
  ggplot2::geom_bar() + scale_fill_aelab_d("control")

## End(Not run)
```

sig_labels *sig_labels*

Description

Run ks_test separately for each level of a faceting variable and return compact letter display (CLD) annotations ready for geom_text. Only facets where the Kruskal-Wallis p-value is below alpha are included in the output.

Usage

```
sig_labels(
  stat_data,
  variable,
  group,
  by = "year",
  plot_data = NULL,
  alpha = 0.05
)
```

Arguments

stat_data	Data frame of raw observations used for the statistical test. Should already be filtered to the relevant subset (e.g. a single water type).
variable	Name of the response variable column (string).
group	Name of the grouping column (string).
by	Name of the faceting column whose levels are iterated over (string, default "year").
plot_data	Data frame used to compute label y-positions via max(variable) per group. Defaults to stat_data. Pass the aggregated/boxplot data here when it differs from the raw test data.
alpha	Significance threshold; facets with KW p-value >= alpha are dropped (default 0.05).

Value

A data frame with columns <group>, Letter, MonoLetter, y_pos, and <by>. Returns an empty data frame when no facet reaches significance.

Examples

```
set.seed(1)
df <- data.frame(
  year = rep(c("2023", "2024"), each = 20),
  grp  = rep(c("A","B","C","D"), 10),
  val  = c(rnorm(20, mean = rep(c(1,3,2,6), 5)), rnorm(20))
)
sig_labels(df, "val", "grp", by = "year")
```

---

tidy_ghg_analyzer *tidy_ghg_analyzer*

Description

Tidy the data downloaded from GHG Analyzer.

Usage

```
tidy_ghg_analyzer(file_path, gas, analyzer = "licor")
```

Arguments

file_path	Directory of file.
gas	Choose between CO2/CH4 or N2O LI-COR Trace Gas Analyzer, which is "ch4" and "n2o", respectively.
analyzer	The brand of the analyzer which the data was downloaded from.

Value

Return the loaded XLSX file after tidying for further analysis.

Examples

```
ghg_data_path <- system.file("extdata", "ch4.xlsx", package = "aelab")
tidy_ghg_analyzer(ghg_data_path, "ch4")
```

Index