Package 'fluorojip'

April 9, 2026

Type Package

Title Analysis of Chlorophyll a Fluorescence Transient Parameters

Version 0.1.1

Description Computes chlorophyll a fluorescence transient parameters from fluorescence summary data, including minimum and maximum fluorescence, selected transient steps, and area. Provides standard photosynthetic performance indices and fluxes per reaction center and per cross section. Includes helpers to read exported trace tables in supported 'csv' formats and validation workflows based on bundled example files, as well as visualization tools and an interactive 'shiny' interface. The implemented calculations are based on Strasser et al. (2004) <doi:10.1007/978-1-4020-3218-9_12> and Stirbet and Govindjee (2011) <doi:10.1016/j.jphotobiol.2010.12.010>.

License GPL-3

Encoding UTF-8

RoxygenNote 7.3.3

Depends R (>= 4.1.0)

Imports stats, utils, graphics, grDevices, readxl

LazyData true

Suggests knitr, rmarkdown, scatterplot3d, shiny, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

NeedsCompilation no

Author Joao Everthon da Silva Ribeiro [aut, cre],
Toshik Iarley da Silva [aut],
Ronald Maldonado Rodriguez [aut]

Maintainer Joao Everthon da Silva Ribeiro <everthonribeiro10@gmail.com>

Repository CRAN

Date/Publication 2026-04-09 08:50:14 UTC

Contents

`calc_fluorojip` *Compute chlorophyll a fluorescence OJIP / JIP-test parameters*

Description

Computes chlorophyll a fluorescence OJIP / JIP-test parameters from fluorescence summary data.

Usage

```
calc_fluorojip(df)
```

Arguments

`df`	A data frame containing fluorescence summary columns. At minimum, provide Fo (`fo` or `o`), Fm (`fm` or `p`), J (`j` or `fj`), and I (`i` or `fi`). Reliable `Mo`, `N`, `PI_abs`, and RC-based fluxes require a K-step / 300 us-equivalent measurement supplied as `k`, `fk`, `f300`, or `f300us`.

Details

The current implementation follows a summary-input workflow. Primary terms such as `phi_Po`, `Vj`, `Vi`, `psi_Eo`, `phi_Eo`, `Mo`, and `PI_abs` are computed explicitly from the supplied fluorescence summary values.

`Mo` is derived from the K-step / F300 region using the standard relation `4 * (F300 - Fo) / (Fm - Fo)`. If a K-step / 300 us-equivalent value is not available, the function does not attempt to guess `Mo`; instead, `Mo`, `N`, RC-based fluxes, and `PI_abs` are returned as `NA` with a warning.

The cross-section outputs `ABS_CSm`, `TRo_CSm`, `ETo_CSm`, and `DIo_CSm` are returned as operational package outputs for internal comparison. They are computed consistently from the supplied summary data, but they should be described carefully if compared against instrument-specific phenomenological flux conventions.

Value

A data frame containing the original input columns plus calculated FluorOJIP / JIP-test parameters.

Examples

```
df <- data.frame(
  sample_id = c("S1", "S2"),
  fo = c(280, 300),
  fm = c(1200, 1250),
  j = c(700, 730),
  i = c(950, 980),
  k = c(340, 360),
  area = c(32000, 35000)
)

res <- calc_fluorojip(df)
res[, c("sample_id", "Fv_Fm", "PI_abs")]
```

`calc_fluorojip_file` *Read a summary table and compute FluorOJIP parameters*

Description

Read a summary table and compute FluorOJIP parameters

Usage

```
calc_fluorojip_file(file, sep = ";", dec = ".", ...)
```

Arguments

`file`	Path to the input summary table.
`sep`	Field separator used in the input file. Defaults to `";"`.
`dec`	Decimal mark used in the input file. Defaults to `"."`.
`...`	Additional arguments passed to `utils::read.csv()`.

Details

This is a convenience wrapper for summary-input workflows: `read.csv(file, ...)` -> `calc_fluorojip(df)`.

Value

A data frame containing the original input columns plus calculated FluorOJIP / JIP-test parameters.

Examples

```
df <- data.frame(
  sample_id = c("S1", "S2"),
  fo = c(280, 300),
  fm = c(1200, 1250),
  j = c(700, 730),
  i = c(950, 980),
  k = c(340, 360),
  area = c(32000, 35000)
)

f <- tempfile(fileext = ".csv")
utils::write.csv(df, f, row.names = FALSE)

res <- calc_fluorojip_file(f, sep = ",")
res[, c("sample_id", "Fv_Fm", "PI_abs")]
```

---

calc_fluorojip_fluorpen

Calculate FluorOJIP parameters from a FluorPen workbook

Description

Calculate FluorOJIP parameters from a FluorPen workbook

Usage

```
calc_fluorojip_fluorpen(file, sheet = 1)
```

Arguments

`file`	Path to the FluorPen `.xlsx` file.
`sheet`	Sheet name or numeric index. Defaults to the first sheet.

Details

This is a convenience wrapper for the typical FluorPen workflow: `read_fluorpen_xlsx(file)` -> `fluorpen_to_ojip(raw)` -> `calc_fluorojip(ojip)`.

Use this function when you want to import a supported FluorPen workbook and compute JIP-test parameters in one step.

Value

A data frame with FluorOJIP / JIP-test parameters calculated from the imported workbook.

Examples

```
xlsx <- system.file("extdata", "FluorPen_test.xlsx", package = "fluorojip")
if (nzchar(xlsx)) {
  res <- calc_fluorojip_fluorpen(xlsx)
  head(res[c("sample_id", "Fv_Fm")])
}
```

`calc_fluorojip_handypea` *Calculate FluorOJIP parameters from a supported exported table*

Description

Reads a supported exported trace table, derives an OJIP summary, and then computes FluorOJIP parameters.

Usage

```
calc_fluorojip_handypea(file)
```

Arguments

`file`	Path to the supported Biolyzer-exported CSV trace table.

Details

Although the function name retains the historical handypea prefix for backward compatibility, the supported import workflow is based on Biolyzer-exported trace tables.

Typical workflow:
`read_handypea_csv()` -> `handypea_to_ojip()` -> `calc_fluorojip()`.

Value

A data frame with FluorOJIP parameters returned by `calc_fluorojip()`.

Examples

```
txt <- c(
  '"Record No",0.00002,0.00027,0.002,0.03,0.10',
  '"S1",280,340,700,950,1200',
  '"S2",300,360,730,980,1250'
)
f <- tempfile(fileext = ".csv")
writeLines(txt, f)

res <- calc_fluorojip_handypea(f)
res[, c("sample_id", "Fv_Fm", "PI_abs")]
```

`example_fluorojip` *Example OJIP summary data*

Description

A dataset containing fluorescence summary values used to demonstrate the calculation of FluorOJIP / JIP-test parameters.

Usage

```
data(example_fluorojip)
```

Format

A data frame with columns for sample identification, treatment groups, and fluorescence summary values used in JIP-test calculations.

Details

The data are organized in the format expected by `calc_fluorojip()`, including sample identifiers, treatment groups, and summary fluorescence variables such as `fo`, `fm`, `j`, `i`, `area`, and, when available, `k`.

Examples

```
data(example_fluorojip)
head(example_fluorojip)
```

`fluorojip_example_biolyzer_file`
Get the bundled Biolyzer validation workbook path

Description

Returns the full path to the example Biolyzer workbook distributed with the package in `inst/extdata`. This file can be used in validation-oriented workflows that compare FluorOJIP outputs against vendor-calculated JIP-test parameters.

Usage

```
fluorojip_example_biolyzer_file()
```

Details

The bundled workbook is intended as a reproducible validation resource for supported Biolyzer-based workflows.

Value

A length-1 character vector containing the normalized full file path.

Examples

```
x <- fluorojip_example_biolyzer_file()
file.exists(x)
basename(x)
```

`fluorpen_to_ojip` *Convert FluorPen traces to an OJIP summary table*

Description

Convert FluorPen traces to an OJIP summary table

Usage

```
fluorpen_to_ojip(x)
```

Arguments

`x`	A list returned by `read_fluorpen_xlsx()`.

Details

This helper converts the FluorPen trace matrix into the fluorescence summary structure expected by the core calculation workflow, including the O, K, J, I, P, and area-related quantities extracted from the trace.

A typical workflow is: `read_fluorpen_xlsx(file)` -> `fluorpen_to_ojip(raw)` -> `calc_fluorojip(ojip)`.

Value

A data frame containing one row per sample with OJIP summary values ready for `calc_fluorojip()`.

Examples

```
xlsx <- system.file("extdata", "FluorPen_test.xlsx", package = "fluorojip")
if (nzchar(xlsx)) {
  raw <- read_fluorpen_xlsx(xlsx)
  ojip <- fluorpen_to_ojip(raw)
  head(ojip)
}
```

handypea_to_ojip *Convert a supported CSV trace export to an OJIP summary table*

Description

Converts a supported exported trace table into an OJIP summary table ready for downstream FluorOJIP parameter calculation.

Usage

```
handypea_to_ojip(x)
```

Arguments

x	A list object returned by `read_handypea_csv()`.

Details

Although the function name retains the historical handypea prefix for backward compatibility, the supported import workflow is based on Biolyzer-exported trace tables.

Time values are normalized internally to milliseconds so the function can target the standard OJIP steps even when exported trace times are recorded in seconds, milliseconds, or microseconds.

Value

A data frame containing `sample_id`, `t_fm`, `fo`, `k`, `fm`, `j`, `i`, `p`, and `area`.

Examples

```
raw <- list(
  times_s = c(0.00002, 0.00027, 0.002, 0.03, 0.10),
  mat = matrix(
    c(
      280, 340, 700, 950, 1200,
      300, 360, 730, 980, 1250
    ),
    nrow = 2,
    byrow = TRUE,
    dimnames = list(c("S1", "S2"), NULL)
  )
)

ojip <- handypea_to_ojip(raw)
ojip
```

normalized_jiptable *Build a normalized JIP parameter table*

Description

Creates a wide or long table of FluorOJIP / JIP-test parameters for each sample, with optional normalization for exploratory analysis, comparison across treatments, and export.

Usage

```
normalized_jiptable(
  df,
  params = NULL,
  sample_col = "sample_id",
  group_col = "treatment",
 normalize = c("none", "zscore", "minmax", "control_ratio", "control_then_zscore"),
  control_level = NULL,
  output = c("wide", "long"),
  digits = 6
)
```

Arguments

df	A data frame, typically the output of calc_fluorojip().
params	Character vector of parameter names to include. If NULL, a default set of commonly used JIP-test parameters is selected.
sample_col	Name of the sample identifier column.
group_col	Name of the grouping column, typically a treatment column.
normalize	Normalization method. One of "none", "zscore", "minmax", "control_ratio", or "control_then_zscore".
control_level	Level of group_col to be used as the control when a control-based normalization method is requested.
output	Output format: "wide" or "long".
digits	Number of decimal places used to round normalized parameter columns.

Details

This function is intended for exploratory analysis and reporting workflows in which selected FluorOJIP / JIP-test parameters need to be compared across samples or treatments on a common scale.

Parameters such as PI_abs, Mo, and RC-based fluxes depend on the availability of a K-step / 300 us-equivalent input (for example, k or f300us) in the original summary data used by calc_fluorojip().

Cross-section outputs such as ABS_CSm, TRo_CSm, ETo_CSm, and DIo_CSm are package outputs intended for operational comparison workflows and may not fully match every instrument-specific convention.

Value

A data frame in wide or long format containing the selected JIP-test parameters after the requested normalization step.

Examples

```
df <- data.frame(
  sample_id = c("S1", "S2", "S3"),
  treatment = c("control", "stress", "stress"),
  fo = c(280, 300, 295),
  fm = c(1200, 1250, 1230),
  j = c(700, 730, 720),
  i = c(950, 980, 970),
  k = c(340, 360, 350),
  area = c(32000, 35000, 34000)
)

res <- calc_fluorojip(df)

tab_wide <- normalized_jiptable(
  res,
  params = c("Fv_Fm", "PI_abs", "ABS_RC"),
  normalize = "zscore",
  output = "wide"
)
tab_wide

tab_long <- normalized_jiptable(
  res,
  params = c("Fv_Fm", "PI_abs"),
  normalize = "control_ratio",
  control_level = "control",
  output = "long"
)
head(tab_long)
```

---

plot_3d_fluorojip *Plot a 3D scatter plot of OJIP parameters*

Description

Creates an exploratory 3D scatter plot for exactly three selected FluorOJIP / JIP-test parameters.

Usage

```
plot_3d_fluorojip(
  res,
  params = c("Fv_Fm", "PI_abs", "area"),
  group_col = "treatment",
```

```
  normalize = TRUE
)

plot_param_3d(
  res,
  params = c("Fv_Fm", "PI_abs", "area"),
  group_col = "treatment",
  normalize = TRUE
)

plot_param_surface3d(
  res,
  params = c("Fv_Fm", "PI_abs", "area"),
  group_col = "treatment",
  normalize = TRUE
)
```

Arguments

res	Data frame with OJIP / JIP-test results.
params	Character vector of exactly 3 parameters to plot.
group_col	Name of the column with treatment or grouping labels.
normalize	Logical indicating whether selected parameters should be z-score normalized before plotting.

Details

`plot_param_3d()` and `plot_param_surface3d()` are aliases of `plot_3d_fluorojip()`. The name `plot_param_surface3d()` is retained for backward compatibility, although the function produces a 3D scatter plot rather than an interpolated surface.

Value

Invisibly returns the object produced by `scatterplot3d::scatterplot3d()`.

Examples

```
if (requireNamespace("scatterplot3d", quietly = TRUE)) {
  df <- data.frame(
    sample_id = c("S1", "S2", "S3"),
    treatment = c("control", "stress", "stress"),
    fo = c(280, 300, 295),
    fm = c(1200, 1250, 1230),
    j = c(700, 730, 720),
    i = c(950, 980, 970),
    k = c(340, 360, 350),
    area = c(32000, 35000, 34000)
  )

  res <- calc_fluorojip(df)
```

```
  plot_3d_fluorojip(
    res,
    params = c("Fv_Fm", "PI_abs", "ABS_RC"),
    group_col = "treatment",
    normalize = TRUE
  )
}
```

`plot_heatmap_fluorojip` *Plot a heatmap of OJIP parameters*

Description

Creates a heatmap for selected FluorOJIP / JIP-test parameters across samples, with optional normalization.

Usage

```
plot_heatmap_fluorojip(
  res,
  params,
  sample_col = "sample_id",
  group_col = "treatment",
  scale = "zscore",
  main = "Normalized JIP-test parameter heatmap"
)

plot_param_heatmap(
  res,
  params,
  sample_col = "sample_id",
  group_col = "treatment",
  scale = "zscore",
  main = "Normalized JIP-test parameter heatmap"
)
```

Arguments

`res`	Data frame with OJIP / JIP-test results.
`params`	Character vector of parameters to plot.
`sample_col`	Name of the column with sample IDs.
`group_col`	Name of the column with treatment or grouping labels.
`scale`	Normalization method. Supported values are `"zscore"`, `"none"`, and `"control_then_zscore"`.
`main`	Title of the plot.

Value

Invisibly returns the object produced by `stats::heatmap()`.

Examples

```
df <- data.frame(
  sample_id = c("S1", "S2", "S3"),
  treatment = c("control", "stress", "stress"),
  fo = c(280, 300, 295),
  fm = c(1200, 1250, 1230),
  j = c(700, 730, 720),
  i = c(950, 980, 970),
  k = c(340, 360, 350),
  area = c(32000, 35000, 34000)
)

res <- calc_fluorojip(df)
plot_heatmap_fluorojip(
  res,
  params = c("Fv_Fm", "PI_abs", "ABS_RC"),
  group_col = "treatment",
  scale = "zscore"
)
```

---

read_fluorpen_xlsx *Read a FluorPen Excel export*

Description

Reads a FluorPen `.xlsx` workbook exported in a wide vendor layout, where columns represent measurements, the first column stores the OJIP time grid, and footer rows contain vendor-calculated JIP-test parameters.

Usage

```
read_fluorpen_xlsx(file, sheet = 1)
```

Arguments

`file`	Path to the FluorPen `.xlsx` file.
`sheet`	Sheet name or numeric index. Defaults to the first sheet.

Details

This function reads a supported FluorPen workbook and prepares its trace and footer summary data for downstream OJIP conversion, validation, and comparison workflows.

A typical workflow is: `read_fluorpen_xlsx(file)` -> `fluorpen_to_ojip(raw)` -> `calc_fluorojip(ojip)`.

Value

A list with metadata, the raw trace matrix, and the vendor summary:

- `sample_id` measurement identifiers taken from the header row
- `measurement_time` acquisition timestamps
- `protocol_id` protocol labels such as `OJIP`
- `times_us` raw time grid in microseconds
- `times_ms` raw time grid converted to milliseconds
- `mat` numeric matrix with one row per sample and one column per time
- `summary_raw` footer summary as imported from the workbook
- `summary_numeric` footer summary converted to numeric values where possible

Examples

```
xlsx <- system.file("extdata", "FluorPen_test.xlsx", package = "fluorojip")
if (nzchar(xlsx)) {
  raw <- read_fluorpen_xlsx(xlsx)
  names(raw)
  dim(raw$mat)
}
```

---

read_handypea_csv *Read a supported Biolyzer-exported CSV trace table*

Description

Reads a CSV trace table exported by Biolyzer and prepares it for downstream OJIP summary extraction.

Usage

```
read_handypea_csv(file)
```

Arguments

`file`	Path to the supported Biolyzer-exported CSV trace table.

Details

Although the function name retains the historical `handypea` prefix for backward compatibility, this workflow is based on supported exported trace tables rather than direct parsing of proprietary raw instrument files.

The function locates the trace header, extracts the time values, reads the numeric trace block, removes trailing non-data rows, and returns the result in a format suitable for `handypea_to_ojip()`.

Value

A list with three elements: `times_s`, the extracted time points; `ids`, the trace identifiers; and `mat`, a numeric matrix containing the fluorescence traces.

Examples

```
txt <- c(
  '"Record No",0.00002,0.00027,0.002,0.03,0.10',
  '"S1",280,340,700,950,1200',
  '"S2",300,360,730,980,1250'
)
f <- tempfile(fileext = ".csv")
writeLines(txt, f)

raw <- read_handypea_csv(f)
raw$times_s
dim(raw$mat)
```

`run_fluorojip_app` *Run the fluorojip Shiny application*

Description

Launches the bundled Shiny interface for interactive FluorOJIP / JIP-test workflows, including data import, parameter calculation, validation, visualization, normalization, and export.

Usage

```
run_fluorojip_app(...)
```

Arguments

`...`	Additional arguments passed to `shiny::runApp()`.

Details

The bundled Shiny application provides an interactive environment for working with fluorescence summary inputs and supported import workflows. Depending on the installed app version, available features may include OJIP curve inspection, parameter selection, normalized 2D plots, heatmaps, 3D plots, validation tabs, export tools, and built-in help content.

This function starts the application distributed with the package and passes any additional arguments directly to `shiny::runApp()`.

Value

This function is called for its side effect of launching the bundled Shiny application.

Examples

```
app_dir <- system.file("shiny", "fluorojip-app", package = "fluorojip")
dir.exists(app_dir)

if (interactive()) {
  run_fluorojip_app()
}
```

`write_normalized_jiptable` *Write a normalized JIP parameter table*

Description

Exports a normalized FluorOJIP / JIP-test parameter table to disk for use in spreadsheet software or downstream statistical workflows.

Usage

```
write_normalized_jiptable(df, file, ...)
```

Arguments

`df`	A data frame to export, typically generated by `normalized_jiptable()`.
`file`	Path to the output file.
`...`	Additional arguments passed to `utils::write.table()`.

Details

The table is written with a semicolon field separator (`sep = ";"`) and decimal point (`dec = "."`). This behavior is intentional and may be convenient for spreadsheet workflows, but it differs from the default behavior of `write.csv()`.

Value

This function is called for its side effect of writing a file to disk. It invisibly returns the output path.

Examples

```
df <- data.frame(
  sample_id = c("S1", "S2"),
  treatment = c("control", "stress"),
  fo = c(280, 300),
  fm = c(1200, 1250),
  j = c(700, 730),
  i = c(950, 980),
  k = c(340, 360),
  area = c(32000, 35000)
```

```
)

res <- calc_fluorojip(df)
tab <- normalized_jiptable(
  res,
  params = c("Fv_Fm", "PI_abs"),
  output = "wide"
)

f <- tempfile(fileext = ".csv")
write_normalized_jiptable(tab, f)
file.exists(f)
```

Index