Package 'CBASSED50'

May 7, 2026

Type Package

Title Process CBASS-Derived PAM Data

Version 0.2.0

Maintainer Luigi Colin <reefgenomics@gmail.com>

Description Tools to process CBASS-derived PAM data efficiently. Minimal requirements are PAM-based photosynthetic efficiency data (or data from any other continuous variable that changes with temperature, e.g. relative bleaching scores) from 4 coral samples (nubbins) subjected to 4 temperature profiles of at least 2 colonies from 1 coral species from 1 site. Please refer to the following CBASS (Coral Bleaching Automated Stress System) papers for in-depth information regarding CBASS acute thermal stress assays, experimental design considerations, and ED5/ED50/ED95 thermal parameters: Nicolas R. Evensen et al. (2023) <doi:10.1002/lom3.10555> Christian R. Voolstra et al. (2020) <doi:10.1111/gcb.15148> Christian R. Voolstra et al. (2025) <doi:10.1146/annurev-marine-032223-024511>.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 7.3.2

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

Imports drc, rlog, stats, utils, dplyr, ggplot2, readxl, glue

Depends R (>= 3.5.0)

Date 2025-05-28

NeedsCompilation no

Author Yulia Iakovleva [aut] (ORCID: <https://orcid.org/0000-0002-0202-7245>), Luigi Colin [aut, cre] (ORCID: <https://orcid.org/0000-0002-6966-2817>), Christian Robert Voolstra [aut] (ORCID: <https://orcid.org/0000-0003-4555-3795>)

Repository CRAN

Date/Publication 2025-06-06 12:50:08 UTC

Contents

`calculate_eds` *Calculate ED5, ED50, and ED95 values for all samples in the dataset.*

Description

Calculate ED5, ED50, and ED95 values for all samples in the dataset.

Usage

```
calculate_eds(
  cbass_dataset,
  grouping_properties = c("Site", "Condition", "Species", "Timepoint"),
  drm_formula = "Pam_value ~ Temperature"
)
```

Arguments

`cbass_dataset`	A data frame containing the dataset to be processed.
`grouping_properties`	A character vector of column names to be used for grouping. Default: c("Site", "Condition", "Species", "Timepoint").

`drm_formula` A formula object specifying the dose-response model. Default: "Pam_value ~ Temperature".

Value

A data frame with ED5, ED50, and ED95 values for each grouping property.

Examples

```
# Example dataset
data(cbass_dataset)

# Extract the ED5, ED50, and ED95 values as a data frame
eds_df <- calculate_eds(cbass_dataset)
```

---

cbass_dataset *CBASS Dataset*

Description

A dataset containing example simulated experimental data.

Usage

```
data(cbass_dataset)
```

Format

A data frame with 240 observations and 9 variables:

Project Name to identify the project/experiment.

Latitude Latitude of the observation collection site in decimal format.

Longitude Longitude of the observation collection site in decimal format.

Date Date of the observation in YYYYMMDD format.

Country Country of the observation in 3-letter ISO country code format.

Site Site of the observation, e.g., name of the reef.

Condition Descriptor to set apart samples from the same species and site, e.g. probiotic treatment vs. control; nursery vs. wild; diseased vs. healthy; can be used to designate experimental treatments besides heat stress. If no other treatments, write 'not available'.

Species Species of the observation. We recommend providing the name of the coral as accurate as possible, e.g. *Porites lutea* or *Porites* sp.

Genotype Denotes samples/fragments/nubbins from distinct colonies in a given dataset; we recommend to use integers, i.e. 1, 2, 3, 4, 5, etc.

Temperature CBASS treatment temperatures; must be ≥ 4 different temperatures; must be integer; e.g. 30, 33, 36, 39. Typical CBASS temperature ranges are average summer mean MMM, MMM+3°C, MMM+6°C, MMM+9°C).

Timepoint Timepoint of PAM measurements in minutes from start of the thermal cycling profile; typically: 420 (7 hours after start, i.e., after ramping up, heat-hold, ramping down) or 1080 (18 hours after start, i.e. at the end of the CBASS thermal cycling profile); differences in ED50s between timepoints 420 and 1080 may be indicative of resilience/recovery (if 1080 ED50 > 420 ED50) or collapse (if 1080 ED50 < 420 ED50).

Pam_value Fv/Fm value of a given sample (format: $\geq$0 and $\leq$1, e.g. 0.387); note that technically any continuous variable can be used for ED50 calculation (e.g., coral whitening; black/white pixel intensity; etc.) and be provided in this column.

Details

This dataset provides experimental data with various attributes for demonstration and testing purposes.

Examples

```
# Load the sample dataset
data(cbass_dataset)
head(cbass_dataset)
```

---

check_enough_unique_temperatures_values

Check if the dataset contains enough unique temperature values

---

Description

Check if the dataset contains enough unique temperature values

Usage

```
check_enough_unique_temperatures_values(dataset)
```

Arguments

`dataset`	The input dataset containing the 'Temperature' column to be analyzed. The 'Temperature' column should be numeric representing temperature values.

Value

Logical value (TRUE or FALSE) indicating whether there are enough unique temperature values (at least 4) in the dataset.

Examples

```
data <- data.frame(Temperature = c(25, 30, 25, 35, 28, 28))
check_enough_unique_temperatures_values(data)
# Output: TRUE
```

convert_columns *Check and Convert Columns in Dataset*

Description

This function checks the data types of specific columns in the provided dataset and converts them to the desired data type if necessary. The function is designed to ensure that certain columns are represented as factors or numeric values, as required for further analysis.

Usage

```
convert_columns(dataset)
```

Arguments

dataset	A data frame containing the dataset to be checked and modified.

Value

A modified version of the input dataset with the specified columns converted to factors or numeric types, as appropriate.

Examples

```
# Sample dataset
data <- data.frame(
                Project = c("Project A", "Project A", "Project B"),
                Date = c("2023-07-31", "2023-08-01", "2023-08-02"),
                Site = c("Site X", "Site Y", "Site Z"),
                Country = c("Country A", "Country B", "Country C"),
                Latitude = c(34.05, 36.16, 40.71),
                Longitude = c(-118.24, -115.15, -74.01),
                Species = c("Species 1", "Species 2", "Species 3"),
                Genotype = c("Genotype A", "Genotype B", "Genotype C"),
                Condition = c("Condition 1", "Condition 2", "Condition 3"),
                Timepoint = c("Timepoint 1", "Timepoint 1", "Timepoint 2"),
                Temperature = c(30.2, 31.5, 29.8),
                Pam_value = c(0.5, 0.6, 0.8)
                )

# Convert columns in the dataset
modified_data <- convert_columns(data)

# The 'Site', 'Condition', 'Species', and 'Genotype' columns are now factors,
# and 'Temperature' and 'PAM' columns are now numeric in the modified_data.
```

dataset_has_mandatory_columns *Check if the dataset has all mandatory columns*

Description

This function checks if a given dataset contains all the mandatory columns specified in the `mandatory_columns` vector.

Usage

```
dataset_has_mandatory_columns(dataset)
```

Arguments

`dataset`	A data frame representing the dataset to be validated.

Value

A logical value indicating whether all the mandatory columns are present in the dataset.

Examples

```
# Sample dataset
sample_data <- data.frame(
                Project = c("Project A", "Project A", "Project B"),
                Date = c("2023-07-31", "2023-08-01", "2023-08-02"),
                Site = c("Site X", "Site Y", "Site Z"),
                Country = c("Country A", "Country B", "Country C"),
                Latitude = c(34.05, 36.16, 40.71),
                Longitude = c(-118.24, -115.15, -74.01),
                Species = c("Species 1", "Species 2", "Species 3"),
                Genotype = c("Genotype A", "Genotype B", "Genotype C"),
                Condition = c("Condition 1", "Condition 2", "Condition 3"),
                Timepoint = c("Timepoint 1", "Timepoint 1", "Timepoint 2"),
                Temperature = c(30.2, 31.5, 29.8),
                Pam_value = c(0.5, 0.6, 0.8)
                )

dataset_has_mandatory_columns(sample_data)
# Output: TRUE

# Sample dataset with missing columns
missing_columns_data <- data.frame(Label = c("A", "B", "C"),
                                   Date = c("2023-07-31", "2023-08-01", "2023-08-02"))

dataset_has_mandatory_columns(missing_columns_data)
# Output: FALSE
```

define_grouping_property *Define Grouping Property Column*

Description

This function creates a new column 'GroupingProperty' in the provided dataset by merging specified columns using a specified separator. The new column is created as a factor variable.

Usage

```
define_grouping_property(dataset, grouping_properties, sep = "_")
```

Arguments

dataset	A data frame where the new 'GroupingProperty' column will be added.
grouping_properties	A character vector containing the names of the columns to be merged.
sep	A character string used as a separator when merging columns (default is "_").

Value

A data frame with the added 'GroupingProperty' column.

Examples

```
# Create a sample data frame
data <- data.frame(Category = c("A", "B", "C"),
                   Subcategory = c("X", "Y", "Z"),
                   Value = c(10, 20, 30))

# Define grouping property using 'Category' and 'Subcategory'
new_data <- define_grouping_property(data, c("Category", "Subcategory"), sep = "-")

# Resulting data frame:
#   Category Subcategory Value GroupingProperty
# 1        A           X    10              A-X
# 2        B           Y    20              B-Y
# 3        C           Z    30              C-Z
```

define_temperature_ranges *Define Temperature Ranges*

Description

This function takes a vector of temperatures and generates a sequence of temperature values that span the range of input temperatures.

Usage

```
define_temperature_ranges(temperatures, n = 100)
```

Arguments

temperatures	A numeric vector containing temperature values.
n	An integer specifying the length of the output sequence. Default is 100.

Value

A numeric vector containing a sequence of temperature values ranging from the minimum temperature to the maximum temperature plus 1, evenly spaced based on the specified length.

Examples

```
temperatures <- c(10, 15, 20, 25, 30)
define_temperature_ranges(temperatures)
```

exploratory_tr_curve *Plot an exploratory temperature response curve.*

Description

Plot an exploratory temperature response curve.

Usage

```
exploratory_tr_curve(
  cbass_dataset,
  grouping_properties = c("Site", "Condition", "Species", "Timepoint"),
  faceting = "Species ~ Site ~ Condition",
  size_text = 12,
  size_points = 2
)
```

Arguments

cbass_dataset	A data frame containing the dataset to be processed.
grouping_properties	A character vector of column names to be used for grouping. Default: c("Site", "Condition", "Species", "Timepoint").
faceting	A formula specifying the faceting of the plot. Default: "Species ~ Site ~ Condition".
size_text	Default: 12. A formula specifying the faceting of the plot.
size_points	Default: 2. A formula specifying the faceting of the plot.

Value

A ggplot object representing the temperature response curve.

Examples

```
# Load example dataset
data(cbass_dataset)

# Create an exploratory temperature response curve
exploratory_curve <- exploratory_tr_curve(cbass_dataset)
```

---

fit_curve_eds *Fit dose-response models and calculate summary statistics for ED5, ED50, and ED95 values.*

Description

Fit dose-response models and calculate summary statistics for ED5, ED50, and ED95 values.

Usage

```
fit_curve_eds(
  cbass_dataset,
  grouping_properties = c("Site", "Condition", "Species", "Timepoint"),
  drm_formula = "Pam_value ~ Temperature"
)
```

Arguments

cbass_dataset	A data frame containing the dataset to be processed.
grouping_properties	A character vector of column names to be used for grouping. Default: c("Site", "Condition", "Species", "Timepoint").
drm_formula	A formula object specifying the dose-response model. Default: "Pam_value ~ Temperature".

Value

A data frame with summary statistics for ED5, ED50, and ED95 values.

Examples

```
# Example dataset
data(cbass_dataset)

# Example grouping properties
grouping_properties <- c("Site", "Condition", "Species", "Timepoint")

# Extract the ED5, ED50, and ED95 values as a data frame
fitted_edss_df <- fit_curve_eds(cbass_dataset, grouping_properties)
```

---

fit_drms *Fit Dynamic Regression Models (DRMs)*

Description

This function fits dynamic regression models (DRMs) to a given dataset using the specified grouping properties and DRM formula.

Usage

```
fit_drms(
  dataset,
  grouping_properties,
  drm_formula,
  is_curveid = FALSE,
  LL.4 = FALSE
)
```

Arguments

dataset	A data frame containing the dataset on which to fit the DRMs.
grouping_properties	A character vector specifying the names of columns in the dataset that will be used as grouping properties for fitting separate DRMs.
drm_formula	A formula specifying the dynamic regression model to be fitted. This formula should follow the standard R formula syntax (e.g., y ~ x1 + x2).
is_curveid	A boolean value indicating if you want to use this parameter to fit the model
LL.4	Logical. If TRUE, the LL.4 model is used instead of LL.3. LL.3 is preferred, as PAM data is expected to never be lower than zero. In cases with overly correlated data and steep slopes, LL.4 allows the lower limit to vary, which can help to better fit the model.

Value

A list of fitted DRM models, with each element corresponding to a unique combination of grouping property values.

Examples

```
data(cbass_dataset)
preprocessed_data <- preprocess_dataset(cbass_dataset)
fit_drms(preprocessed_data,
       c("Site", "Condition", "Species", "Genotype"),
        "Pam_value ~ Temperature")
```

---

get_all_eds_by_grouping_property

Get ED5s, ED50s and ED95s by Grouping Property

Description

This function takes a list of models and extracts the ED5s, ED50s and ED95s values for each model based on a specified grouping property. The ED5s, ED50s and ED95s values is extracted from the model's coefficients and is associated with the intercept term.

Usage

```
get_all_eds_by_grouping_property(models)
```

Arguments

`models`	A list of models where each element represents a model object containing coefficients.

Value

A data frame containing the ED50 values along with their corresponding grouping property. Each row represents a model's ED50 value and its associated grouping property.

Examples

```
data(cbass_dataset)
preprocessed_data <- preprocess_dataset(cbass_dataset)
model_list <- fit_drms(preprocessed_data,
           c("Site", "Condition", "Species", "Genotype"),
            "Pam_value ~ Temperature", is_curveid = TRUE)
eds_data <- get_all_eds_by_grouping_property(model_list)

# Resulting data frame structure:
#   ED5         ED50         ED95         GroupingProperty
# 1 ED5_value_1 ED50_value_1 ED95_value_1 Group1
# 2 ED5_value_2 ED50_value_2 ED95_value_2 Group2
```

get_ed50_by_grouping_property *Get ED50 by Grouping Property*

Description

This function takes a list of models and extracts the ED50 value for each model based on a specified grouping property. The ED50 value is extracted from the model's coefficients and is associated with the intercept term.

Usage

```
get_ed50_by_grouping_property(models)
```

Arguments

models	A list of models where each element represents a model object containing coefficients.

Value

A data frame containing the ED50 values along with their corresponding grouping property. Each row represents a model's ED50 value and its associated grouping property.

Examples

```
data(cbass_dataset)
preprocessed_data <- preprocess_dataset(cbass_dataset)
model_list <-fit_drms(preprocessed_data,
           c("Site", "Condition", "Species", "Genotype"),
            "Pam_value ~ Temperature")
ed50_data <- get_ed50_by_grouping_property(model_list)

# Resulting data frame structure:
#   ED50     GroupingProperty
# 1 ED50_value_1 Group1
# 2 ED50_value_2 Group2
```

get_predicted_pam_values *Get Predicted PAM Values*

Description

This function takes a list of models and a temperature range, and generates predicted PAM (Pulse Amplitude Modulation) values based on the provided models and temperature range.

Usage

```
get_predicted_pam_values(models, temp_range)
```

Arguments

models	A list of model objects representing PAM prediction models.
temp_range	A numeric vector containing a sequence of temperature values for which PAM predictions will be generated.

Value

A data frame containing the predicted PAM values along with corresponding temperature values from the given temperature range.

See Also

predict_temperature_values, transform_predictions_to_long_dataframe, define_temperature_ranges
predict_temperature_values transform_predictions_to_long_dataframe define_temperature_ranges

Examples

```
# Load models and temperature range
# To load internal dataset that is provided with the R package
data("cbass_dataset")
cbass_dataset <- preprocess_dataset(cbass_dataset)
grouping_properties <- c("Site", "Condition", "Species", "Timepoint")
drm_formula <- "Pam_value ~ Temperature"

# Make list of model
models <- fit_drms(cbass_dataset, grouping_properties, drm_formula, is_curveid = FALSE)
temp_ranges <- define_temperature_ranges(cbass_dataset$Temperature, n = 100)

# Get predicted Pam_value values
predicted_pam <- get_predicted_pam_values(models, temp_ranges)
```

---

mandatory_columns *Get the names of mandatory columns for the dataset.*

Description

This function returns a character vector containing the names of columns that are considered mandatory for a dataset to meet certain requirements.

Usage

```
mandatory_columns()
```

Value

A character vector containing the names of mandatory columns.

Examples

```
mandatory_cols <- mandatory_columns()
print(mandatory_cols)

# [1] "Project"     "Date"        "Site"        "Genotype"
# [5] "Species"     "Country"     "Latitude"    "Longitude"
# [9] "Condition"   "Temperature" "Timepoint"   "Pam_value"
```

---

plot_ED50_box *Plot a boxplot of ED50 values for different species and conditions.*

Description

Plot a boxplot of ED50 values for different species and conditions.

Usage

```
plot_ED50_box(
  cbass_dataset,
  grouping_properties = c("Site", "Condition", "Species", "Timepoint"),
  drm_formula = "Pam_value ~ Temperature",
  Condition = "Condition",
  faceting = "~ Site",
  size_text = 12,
  size_points = 2
)
```

Arguments

cbass_dataset	A data frame containing the dataset to be processed.
grouping_properties	A character vector of column names to be used for grouping. Default: c("Site", "Condition", "Species", "Timepoint")
drm_formula	A formula object specifying the dose-response model. Default: "Pam_value ~ Temperature".
Condition	A character string specifying the condition to be used for coloring the plot. Default: "Condition".
faceting	A formula specifying the faceting of the plot. Default: "~ Site".
size_text	Default: 12. A formula specifying the faceting of the plot.
size_points	Default: 2. A formula specifying the faceting of the plot.

Value

A ggplot object representing the boxplot of ED50 values.

Examples

```
# Example dataset
data(cbass_dataset)

# Example grouping properties
grouping_properties <- c("Site", "Condition", "Species", "Timepoint")

# Make ggplot object
boxplot_ED50 <- plot_ED50_box(cbass_dataset)
```

`plot_model_curve` *Plot the model curve with predicted PAM values and confidence intervals.*

Description

Plot the model curve with predicted PAM values and confidence intervals.

Usage

```
plot_model_curve(
  cbass_dataset,
  grouping_properties = c("Site", "Condition", "Species", "Timepoint"),
  drm_formula = "Pam_value ~ Temperature",
  faceting_model = "Species ~ Site ~ Condition",
  size_text = 12,
  size_points = 2
)
```

Arguments

`cbass_dataset`	A data frame containing the dataset to be processed.
`grouping_properties`	A character vector of column names to be used for grouping. Default: c("Site", "Condition", "Species", "Timepoint").
`drm_formula`	A formula object specifying the dose-response model. Default: "Pam_value ~ Temperature".
`faceting_model`	A formula specifying the faceting of the plot. Default: "Species ~ Site ~ Condition".
`size_text`	Default: 12. A formula specifying the faceting of the plot.
`size_points`	Default: 2. A formula specifying the faceting of the plot.

Value

A ggplot object representing the model curve with predicted PAM values.

Examples

```
data(cbass_dataset)

model_curve_plot <- plot_model_curve(cbass_dataset)
```

`predict_temperature_values` *Predict the temperature values*

Description

This function takes a list of models and generates a sequence of temperature values that span the range of input temperatures.

Usage

```
predict_temperature_values(models, temp_range)
```

Arguments

`models`	A list of models where each element represents a model object containing coefficients.
`temp_range`	the temperature range to be used for predictions from the function define_temperature_ranges

Value

A data frame containing the predicted PAM values for each temperature along with their corresponding grouping property. Each row represents a model's predicted PAM value and its associated grouping property and confidence interval.

Examples

```
data(cbass_dataset)
preprocessed_data <- preprocess_dataset(cbass_dataset)

models <- fit_drms(preprocessed_data,
                   c("Site", "Condition", "Species", "Timepoint"),
                     "Pam_value ~ Temperature", is_curveid = TRUE)
temp_ranges <- define_temperature_ranges(cbass_dataset$Temperature, n = 100)
predict_temperature_values(models, temp_ranges)
```

preprocess_dataset *Preprocesses the data by converting and checking column data types.*

Description

This function preprocesses the input data by performing checks on column data types and converting them if necessary. It ensures that the dataset meets certain requirements before further analysis or modeling.

Usage

```
preprocess_dataset(dataset)
```

Arguments

dataset	A data frame containing the dataset to be preprocessed.

Value

A preprocessed data frame with converted and validated column data types.

Examples

```
# Load a sample dataset
data(cbass_dataset)
# Preprocess the dataset
preprocessed_data <- preprocess_dataset(cbass_dataset)
```

process_dataset *Experimental Features for CBASS Dataset Analysis*

Description

This script contains functions for processing and analyzing CBASS datasets, including fitting dose-response models, calculating effective doses (ED5, ED50, ED95), and plotting results.

Process the dataset by preprocessing, validating, and defining grouping properties.

Usage

```
process_dataset(cbass_dataset, grouping_properties)
```

Arguments

cbass_dataset	A data frame containing the dataset to be processed.
grouping_properties	A character vector of column names to be used for grouping. Default: c("Site", "Condition", "Species", "Timepoint").

Value

A processed data frame with the grouping property defined.

Examples

```
# Example dataset
data(cbass_dataset)

# Example grouping properties
grouping_properties <- c("Site", "Condition", "Species", "Timepoint")

# Process the dataset
processed_dataset <- process_dataset(cbass_dataset, grouping_properties)
```

---

read_data *Read Data Function*

Description

Reads data from a file based on its format (Excel or CSV).

Usage

```
read_data(file_path)
```

Arguments

file_path	Character string specifying the path to the input file.

Details

This function determines the file format based on the file extension and uses appropriate methods to read data from either Excel (xls, xlsx) or CSV (csv, txt) files.

Value

A data frame containing the read data.

Examples

```
# Read data from an Excel file
read_data(system.file("extdata", "example.xlsx", package = "CBASSED50"))

# Read data from a CSV file
read_data(system.file("extdata", "example.csv", package = "CBASSED50"))
```

transform_predictions_to_long_dataframe

Transform Predictions to a Long-Format DataFrame

Description

This function takes a list of predictions and converts them into a long-format data frame. Each prediction corresponds to a different temperature range.

Usage

```
transform_predictions_to_long_dataframe(predictions, temp_range)
```

Arguments

predictions	A list of data frames where each data frame represents predictions for a specific temperature range. The data frames should have a common grouping property.
temp_range	the temperature range to be used for predictions from the function define_temperature_ranges

Value

A long-format data frame containing the transformed predictions with columns for "GroupingProperty," "Temperature," and "PredictedPAM."

validate_cbass_dataset

Validate CBASS Dataset

Description

This function validates a dataset to ensure it contains all the mandatory columns required for further processing. If any mandatory columns are missing, it raises an error with the list of missing columns.

Usage

```
validate_cbass_dataset(dataset)
```

Arguments

dataset	A data frame representing the CBASS dataset to be processed and validated.

Value

A processed and validated CBASS dataset with appropriate data types for its columns.

See Also

dataset_has_mandatory_columns, convert_columns, check_enough_unique_temperatures_values

Examples

```
# Assuming a dataset named 'cbass_dataset' is available in the environment
data(cbass_dataset)
preprocessed_data <- preprocess_dataset(cbass_dataset)
validate_cbass_dataset(preprocessed_data)
```

Index