Package: microdiluteR (via r-universe)

October 22, 2024

Type Package

Title Analysis of Broth Microdilution Assays

Version 1.0.1

Date 2024-04-16

Description A framework for analyzing broth microdilution assays in various 96-well plate designs, visualizing results and providing descriptive and (simple) inferential statistics (i.e. summary statistics and sign test). The functions are designed to add metadata to 8 x 12 tables of absorption values, creating a tidy data frame. Users can choose between clean-up procedures via function parameters (which covers most cases) or user prompts (in cases with complex experimental designs). Users can also choose between two validation methods, i.e. exclusion of absorbance values above a certain threshold or manual exclusion of samples. A function for visual inspection of samples with their absorption values over time for certain group combinations helps with the decision. In addition, the package includes functions to subtract the background absorption (usually at time T0) and to calculate the growth performance compared to a baseline. Samples can be visually inspected with their absorption values displayed across time points for specific group combinations. Core functions of this package (i.e. background subtraction, sample validation and statistics) were inspired by the manual calculations that were applied in Tewes and Muller (2020) <doi:10.1038/s41598-020-67600-7>.

URL https://silvia-eckert.github.io/microdiluteR/

BugReports https://github.com/silvia-eckert/microdiluteR/issues

License GPL (>=3)

Depends R (>= 4.3)

Imports dplyr (>= 1.1.4), ggh4x (>= 0.2.8), ggplot2 (>= 3.5.0), ggthemes (>= 5.0.1), magrittr (>= 2.0.3), purrr (>= 1.0.2), rlang (>= 1.1.3), rstatix (>= 0.7.2), stringr (>= 1.5.1), tibble (>= 3.2.1), tools (>= 4.3.3), vctrs (>= 0.6.5)

Suggests knitr (>= 1.4.6), rmarkdown (>= 2.26), testthat (>= 3.0.0)

Encoding UTF-8

LazyData true

RoxygenNote 7.3.1

Repository https://silvia-eckert.r-universe.dev

RemoteUrl https://github.com/silvia-eckert/microdiluter

RemoteRef HEAD

RemoteSha 106a132b083c0400f395e54f68877c6eaa32d1d2

Contents

`add_concentration` *Add concentration metadata*

Description

`add_concentration` adds concentration metadata to photometer data that is specified in long format. For this function to work properly, the column containing the well positions should be named 'Position' and the column containing the corresponding absorption values should be named 'Values'.

`generate_concentration_list` generates a list of provided concentration levels mapped to the user-specified plate layout. The plate layout is based on a 96-well plate and can be either horizontal (i.e. letters A-H) or vertical (i.e. numbers 1-12).

`ask_concentration_list` works the same way as `generate_concentration_list`, but retrieves the concentration levels based on a user prompt instead of user-set parameters. The plate axis can

be either in horizontal direction providing letters A-H or in vertical direction providing numbers 1-12 based on a 96-well plate layout.

`match_concentration` maps concentration levels to corresponding well positions and returns 'NA' otherwise.

Usage

```
add_concentration(
  input_data,
  concentration_list = NULL,
  ask_concentration_list = TRUE,
  ...
)

generate_concentration_list(concentration_levels, direction)

ask_concentration_list(direction = c("horizontal", "vertical"))

match_concentration(well_position, concentration_list)
```

Arguments

`input_data`	A data frame with well positions and their corresponding values.
`concentration_list`	A list containing concentration information
`ask_concentration_list`	A boolean parameter indicating whether concentration levels should be retrieved via user prompt (default) or not.
`...`	Additional arguments to be passed to `ask_concentration_list`.
`concentration_levels`	A numeric vector containing concentration levels.
`direction`	A character vector specifying the orientation of the plate layout. It can be either "horizontal" or "vertical".
`well_position`	The sample position(s) to check

Details

`generate_concentration_list` checks if the length of `concentration_levels` matches the specified number of rows or columns based on the direction parameter. If not, it throws an error. If the lengths match, it generates a list of concentration levels where each level is assigned to a corresponding row or column based on the direction parameter.

Value

`add_concentration` returns a data frame with concentration metadata added.

`generate_concentration_list` returns a list of concentration levels where each level is assigned to a corresponding row or column based on the selected direction parameter.

ask_concentration_list returns a list containing plate axes as keys and concentration information as values.

match_concentration returns the corresponding concentration level if sample position matches concentration criteria, "NA" otherwise

See Also

generate_experiment_list, ask_experiment_list, generate_group_list, ask_group_list, add_treatment, validate_cells

---

add_treatment *Add treatment metadata*

Description

add_treatment adds treatment metadata to photometer data that is specified in long format. For this function to work properly, the column containing the well positions should be named 'Position' and the column containing the corresponding absorption values should be named 'Values'.

generate_treatment_list generates a list of provided treatment labels mapped to the user-specified plate layout. The plate layout is based on a 96-well plate and can be either horizontal (i.e. letters A-H) or vertical (i.e. numbers 1-12).

ask_treatment_list works the same way as generate_treatment_list, but retrieves the treatment labels based on a user prompt instead of user-set parameters. The plate axis can be either in horizontal direction providing letters A-H or in vertical direction providing numbers 1-12 based on a 96-well plate layout.

match_treatment maps treatment labels to corresponding well positions and returns 'NA' otherwise.

Usage

```
add_treatment(
  input_data,
  treatment_list = NULL,
  ask_treatment_list = TRUE,
  ...
)

generate_treatment_list(treatment_labels, direction)

ask_treatment_list(direction = c("horizontal", "vertical"))

match_treatment(well_position, treatment_list)
```

Arguments

input_data	A data frame with well positions and their corresponding values.
treatment_list	A list containing treatment information
ask_treatment_list	A boolean parameter indicating whether treatment labels should be retrieved via user prompt (default) or not.
...	Additional arguments to be passed to ask_treatment_list.
treatment_labels	A character vector containing treatment labels.
direction	A character vector specifying the orientation of the plate layout. It can be either "horizontal" or "vertical".
well_position	The sample position(s) to check

Details

generate_treatment_list checks if the length of treatment_labels matches the specified number of rows or columns based on the direction parameter. If not, it throws an error. If the lengths match, it generates a list of treatment_labels where each label is assigned to a corresponding row or column based on the direction parameter.

Value

add_treatment returns a data frame with treatment information added.

generate_treatment_list returns a list of treatment labels where each level is assigned to a corresponding row or column based on the selected direction parameter.

ask_treatment_list returns a list containing plate axes as keys and treatment labels as values.

match_treatment returns the corresponding treatment labels if sample position matches treatment criteria, "NA" otherwise

See Also

generate_experiment_list, ask_experiment_list, generate_group_list, ask_group_list, add_concentration, validate_cells

---

apply_sign_test *Apply sign test*

Description

This function applies the one-sample sign test to input data grouped by specified variables.

Usage

```
apply_sign_test(
  stats_data,
  summarized_data,
  value = "Value",
  p.signif = "p.signif",
  grouping = NULL,
  na = "NA"
)
```

Arguments

stats_data	A data frame containing the calculated growth performance data, e.g. via a function call to calculate_growth_performance.
summarized_data	A data frame containing corresponding summarized data, e.g. via function call summarize_growth_performance.
value	The column containing absorption values to be tested. Defaults to 'Value'.
p.signif	The column containing significance denoted in asterisk notation. Defaults to 'p.signif'.
grouping	A character vector specifying the grouping variables.
na	A character value specifying the keyword to display if sign tests cannot be applied on subsets of the data (e.g. because of too small sample sizes). Defaults to "NA".

Value

A data frame containing the summarized data with sign test results added.

See Also

calculate_growth_performance, summarize_growth_performance, plot_growth_performance,

---

bma *Absorption values from six broth microdilution assays conducted on 96-well plates*

Description

A list of six sample data sets with absorption values from broth microdilution assays on 96-well plates, applied to two groups with one experiment each at two time points T0 and T3.

Usage

```
bma
```

Format

A list with six data frames:

bma_grp1_exp2_T0 Absorption values from a broth microdilution assay applied on group 1 from experiment 2 on 96-well plate at timepoint T0.

bma_grp1_exp2_T3 Absorption values from a broth microdilution assay applied on group 1 from experiment 2 on 96-well plate at timepoint T3.

bma_grp2_exp1_T0 Absorption values from a broth microdilution assay applied on group 2 from experiment 1 on 96-well plate at timepoint T0.

bma_grp2_exp1_T3 Absorption values from a broth microdilution assay applied on group 2 from experiment 1 on 96-well plate at timepoint T3.

Details

The data was derived from two broth microdilution assay experiments testing the growth performance of *Botrytis cinerea* conidia on two *Tanacetum vulgare* chemotypes (defined as `groups`). Leaf extracts from chemotypes were fractionated using solid-phase extraction with a water-methanol polarity gradient (defined as `treatment`) and fractions were subjected to assays in two concentrations (100 ppm and 200 ppm) plus a positive control for each concentration level. The 96-well plate design was assigned in horizontal direction (provided as `plate_axis`) and is stored in the metadata attribute of the list. The data was generated for teaching purposes and is unrestricted by any licensing constraints.

Examples

```
data(bma)
attr(bma, "metadata")
```

---

calculate_growth_performance

Calculate and visualize growth performance

Description

`calculate_growth_performance` standardizes data by subtracting the average value of the control group from each treatment level for each concentration level, applied within each experiment. It assumes the input data is a data frame with columns 'Experiment', 'Concentration', 'Treatment', and 'Value', where 'Concentration' represents different concentration levels, 'Treatment' represents different treatment groups, and 'Value' represents the corresponding absorption values.

`calculate_percentage_change` calculates the percentage change between a vector of values (or a single value) and a reference value as the baseline. If a value in the vector is less than the reference, it returns the negative percentage difference; otherwise, it returns the positive percentage difference.

`summarize_growth_performance` summarizes a data frame containing growth performance by computing the mean and either the standard error or standard deviation.

`plot_growth_performance` visualizes growth performance using bar charts with error bars.

Usage

```
calculate_growth_performance(
  input_data,
  treatment_grouping = FALSE,
  concentration_grouping = FALSE,
  group = "Group",
  experiment = "Experiment",
  treatment = "Treatment",
  concentration = "Concentration",
  timepoint = "Timepoint",
  value = "Value",
  control_mean = "control_mean"
)

calculate_percentage_change(input, reference)

summarize_growth_performance(
  input_data,
  compute_sd = FALSE,
  grouping = c("Group", "Treatment", "Concentration", "Timepoint"),
  treatment = "Treatment",
  value = "Value"
)

plot_growth_performance(
  input_data,
  stats_data = NULL,
  level_unit = NULL,
  treatment_order = NULL,
  apply_sign_test = FALSE,
  grouping = NULL,
  x_var = "Treatment",
  y_var = "mean",
  error_var = "stderr",
  x_lab = "Treatment",
  y_lab = NULL,
  fill_var = "Concentration",
  row_facets = NULL,
  col_facets = "Group",
  value = "Value",
  p_values = "p.signif",
  level_colors = NULL,
  ...
)
```

Arguments

input_data	A data frame containing summarized data, e.g. from function call to summarize_growth_performance.

`treatment_grouping`
: A Boolean value that specifies whether or not (default) there is a treatment grouping within the plate.

`concentration_grouping`
: A Boolean value that specifies whether or not (default) there is a concentration grouping within the plate.

`group`
: The column containing group information. Defaults to 'Group'.

`experiment`
: The column containing experiment information. Defaults to 'Experiment'. The hierarchy is group > experiment, i.e. within a single group, there might be several experiments taking place (e.g. multiple extracts from the same plant species tested with plant species being the group and type of extract being the experiment).

`treatment`
: The column containing treatment information. Defaults to 'Treatment'.

`concentration`
: The column containing concentration information. Defaults to 'Concentration'.

`timepoint`
: The column containing timepoint information. Defaults to 'Timepoint'.

`value`
: The column containing the absorption values to be assessed via `apply_sign_test`. Defaults to 'Value'.

`control_mean`
: he column containing the absorption values to calculate growth performance. Defaults to 'control_mean'.

`input`
: A single numeric value.

`reference`
: A single numeric value serving as the baseline for comparison.

`compute_sd`
: Logical, indicating whether to compute the standard deviation (default) or standard error.

`grouping`
: Optional. A character vector specifying the grouping variables on which to apply the sign test. If not specified and 'apply_sign_test' is set to TRUE, then the test will be applied on the whole dataset.

`stats_data`
: Optional. A data frame containing growth performance data, e.g. from function call to `calculate_growth_performance`. Only necessary, if 'apply_sign_test' parameter is set to TRUE.

`level_unit`
: Optional. The unit of applied concentrations to display on the y-axis.

`treatment_order`
: Optional. An alternative order of factor levels on the x-axis.

`apply_sign_test`
: Logical. Should the sign test be applied to specified levels? For this, the 'stats_data' and 'grouping' parameters need to be specified.

`x_var`
: The variable name for the x-axis. Defaults to "Treatment".

`y_var`
: The variable name for the y-axis. Defaults to "mean".

`error_var`
: The variable name to generate the error bars. Defaults to 'stderr'.

`x_lab`
: The label for the x-axis. Defaults to "Treatment".

`y_lab`
: Optional. The label for the y-axis. If not provided will return "Relative growth performance".

`fill_var`
: The variable used to fill facets. Defaults to "Concentration".

row_facets	A character vector specifying nested column facets. Defaults to NULL.
col_facets	A character vector specifying nested row facets. Defaults to "Group".
p_values	The column containing the (adjusted) p-values. Defaults to 'p.adj.signif' from a function call to apply_sign_test and rstatix::sign_test.
level_colors	Optional. The colors for different levels. If not specified, will be determined based on levels of 'fill_var' using gray.colors.
...	Additional arguments to be passed to apply_sign_test.

Details

plot_growth_performance uses ggplot2 to create bar charts of summarized data with error bars.

Value

calculate_growth_performance returns a modified data frame with the control mean subtracted from each treatment level for each concentration level, applied within each experiment.

calculate_percentage_change returns a numeric vector containing the percentage change for each value in the vector compared to the reference.

summarize_growth_performance returns a data frame containing the summary statistics.

plot_growth_performance returns a ggplot object.

See Also

tidy_single_plate, tidy_plates_via_params, tidy_plates_via_prompts

---

check_well_positions *Check monotonicity of well positions across groups*

Description

check_well_positions checks if well positions across groups, i.e. experiments, monotonically increase or decrease with timepoints measured.

check_monotonicity checks whether the values in a numeric vector are monotonically increasing or decreasing.

Usage

```
check_well_positions(
  input_data,
  x_var = "Timepoint",
  y_var = "Value",
  grouping = "Position",
  v_var = "Validity",
  wp_var = "Position"
)

check_monotonicity(vec)
```

Arguments

input_data	A data.frame containing the input data, e.g. from a function call to tidy_single_plate, tidy_plates_via_params or tidy_plates_via_prompts.
x_var	A character string specifying the variable to be plotted on the x-axis. Defaults to 'Timepoint'.
y_var	A character string specifying the variable to be plotted on the y-axis. Defaults to 'Value'.
grouping	A vector of character strings specifying the grouping variables. Defaults to 'Position' if no grouping is provided.
v_var	A character string specifying the validity information. Usually a column with all rows being 'valid'. Rows are set to 'invalid' based on user selection. Defaults to "Validity".
wp_var	A character string specifying the column providing the well positions. Defaults to "Position".
vec	A numeric vector to be checked for monotonicity.

Details

If non-monotonic groups of well positions are detected, check_well_positions plots them as line graphs and returns a list with both the corresponding subset of the data for further inspection and the input data adjusted for invalid well positions from visual inspection.

check_monotonicity checks if all differences between consecutive elements in the vector 'vec' are non-negative (indicating monotonic non-decreasing behavior) or non-positive (indicating monotonic non-increasing behavior).

Value

check_well_positions returns a subset of the input data containing only the data from non-monotonic groups, if non-monotonic groups are detected. Otherwise, NULL is returned.

check_monotonicity returns a logical value.

See Also

tidy_plate, tidy_plates_via_params, tidy_plates_via_prompts

validate_cells, update_validity

Examples

```
# Generate example data
set.seed(123)
df <- data.frame(Position = rep(1:21, 2),
                 Value = c(1:21, sample(1:21,21, TRUE)),
                 Timepoint = rep(paste0("T",1:3),14),
                 Validity = "valid",
                 Group_1 = rep(LETTERS[1:2], each=21),
                 Group_2 = rep(letters[1:14], each = 3))
# All groups behave monotonically
```

```
check_well_positions(df[df$Group_1 == "A",],
                     x_var = "Timepoint",
                     y_var = "Value",
                     grouping = c("Group_1", "Group_2"))
# Six groups behave non-monotonically
check_well_positions(df[df$Group_1 == "B",],
                     x_var = "Timepoint",
                     y_var = "Value",
                     grouping = c("Group_1", "Group_2"))
# Check if a vector is monotonically increasing (will return TRUE)
check_monotonicity(c(1, 2, 3, 4, 5))
# Check if a vector is monotonically decreasing (will return FALSE)
check_monotonicity(c(5, 80, 3, 2, 1))
```

`generate_experiment_list` *Generate list of experiment names from user parameters*

Description

`generate_experiment_list` generates a list of provided experiment names extracted from file names.

`ask_experiment_list` works the same way as `generate_experiment_list`, but retrieves the experiment names based on a user prompt instead of user-set parameters.

Usage

```
generate_experiment_list(experiment_names, file_list)

ask_experiment_list(file_list)
```

Arguments

`experiment_names`	A character vector containing names for each experiment.
`file_list`	A character vector of file IDs. Used to extract experiment IDs from.

Details

`generate_experiment_list` extracts unique identifiers from file names and matches them with the provided experiment names. If the number of experiment names does not match the number of unique identifiers extracted from the file names, it throws an error. If the lengths match, it generates a list of experiment names where each name is associated with a unique identifier extracted from the file names.

Value

generate_experiment_list returns a list of experiment names where each level is assigned to a corresponding row or column based on the selected direction parameter.

ask_experiment_list returns a list containing experiment identifiers as keys and experiment names as values.

See Also

generate_group_list, ask_group_list, add_treatment, add_concentration

---

generate_group_list *Generate list of group IDs from user parameters*

Description

generate_group_list generates a list of provided group IDs extracted from file IDs.

ask_group_list works the same way as generate_group_list, but retrieves the group IDs based on a user prompt instead of user-set parameters.

Usage

```
generate_group_list(group_names, file_list)

ask_group_list(file_list)
```

Arguments

group_names	A character vector containing IDs for each group.
file_list	A character vector of file IDs. Used to extract group IDs from.

Details

generate_group_list extracts unique identifiers from file IDs and matches them with the provided group IDs. If the number of group IDs does not match the number of unique identifiers extracted from the file IDs, it throws an error. If the lengths match, it generates a list of group IDs where each ID is associated with a unique identifier extracted from the file IDs.

Value

generate_group_list returns a list of group IDs where each level is assigned to a corresponding row or column based on the selected direction parameter.

ask_group_list returns a list containing group identifiers as keys and group IDs as values.

See Also

generate_experiment_list, ask_experiment_list, add_treatment, add_concentration

read_plates *Read multiple text files from photometer measurement*

Description

read_plates reads raw text files generated from photometer measurements of 96-well plates. The data is returned as a list but without additional first lines that are sometimes used to provide additional information, for example, wavelength used or the date of measurement. For comparison, this information is saved as an attribute of the list and can be retrieved via the "info" parameter.

read_plate reads a raw text file generated from a photometer measurement of a 96-well plate. The data is returned as is but without additional first lines that are sometimes used to provide additional information, for example, wavelength used or the date of measurement. For comparison, this information is saved as an attribute of the raw data and can be retrieved via the "info" parameter.

Usage

```
read_plates(input_data, pattern = NULL, skip_lines = 2)

read_plate(file_path, skip_lines = 2)
```

Arguments

input_data	Either a folder path containing raw data files or a list of data frames.
pattern	A character value providing the file pattern to search for. If not provided, defaults to "^BMA|bma".
skip_lines	A numerical value that specifies the number of lines to be skipped until data is provided. These lines will be saved as an attribute and are accessible via the "info" parameter. Defaults to 2.
file_path	The file path to the file containing the raw data.

Value

read_plates returns a list of data frames containing the raw photometer data.

read_plate returns a data frame containing the raw photometer data

subtract_T0 *Subtract timepoint T0 and remove from data*

Description

This function subtracts the values at timepoint T0 from all other timepoints and removes it from the data.

Usage

```
subtract_T0(
  input_data,
  grouping = c("Group", "Experiment", "Position"),
  value = "Value",
  timepoint = "Timepoint",
  validity = "Validity"
)
```

Arguments

input_data	A data frame containing columns preferably named as 'Position', 'Value', 'Experiment','Validity', and 'Timepoint'.
grouping	A character vector specifying the columns to use for grouping. Defaults to c("Experiment", "Position").
value	The column containing the values to be modified. Defaults to "Value".
timepoint	The column containing the timepoint information. Defaults to "Timepoint".
validity	The column containing validity information. Defaults to "Validity".

Details

This function modifies the input data frame by subtracting the value at T0 timepoint from all other timepoints for each plate (i.e. experiment). It then removes the rows with this timepoint from the data frame.

Value

A modified data frame with timepoint T0 subtracted and removed.

---

tidy_plates *Add metadata to values from photometer measurements*

Description

Cleans a list of data frames with different structures. This function reads data from either a folder containing text files or from a list of data frames. It then cleans each data frame using the function `tidy_plates_via_prompts()`.

Usage

```
tidy_plates(
  input_data,
  how_many = c("single", "multiple"),
  user_prompt = FALSE,
  multiple_structures = FALSE,
  direction = c("horizontal", "vertical"),
  ...
)
```

Arguments

input_data	Either a folder path containing text files or a list of data frames.
how_many	A character vector specifying if metadata should be added to only a single plate or multiple plates.
user_prompt	Logical indicating whether adding metadata should be applied via user prompts. Only applied if user_prompt is set to TRUE. Defaults to FALSE.
multiple_structures	Logical indicating whether adding metadata should be applied for each plate separately, because plates have different metadata structures. Will be applied via user prompts for each plate separately. Defaults to FALSE.
direction	A character vector specifying the orientation of the plate layout. It can be either "horizontal" or "vertical".
...	Additional arguments to be passed to read_plates, tidy_single_plate, tidy_plates_via_params, or tidy_plates_via_prompts.

Value

A list of cleaned data frames.

See Also

read_plates, tidy_plates_via_prompts, tidy_plates_via_params, tidy_single_plate

---

tidy_plates_via_params

Tidy multiple 96-well plates via parameters

Description

This function processes raw plates data from photometer measurements, adds metadata via user-specified parameter values, and combines processed data into a single data frame.

Usage

```
tidy_plates_via_params(
  input_data,
  direction = c("horizontal", "vertical"),
  group_IDs = NULL,
  experiment_names = NULL,
  validity_method = c("threshold", "invalid"),
  threshold = NULL,
  invalid_samples = NULL,
  treatment_labels,
  concentration_levels,
  ...
)
```

Arguments

input_data	Either a folder path containing raw data files or a list of data frames.
direction	A character vector specifying the orientation of the plate layout. It can be either "horizontal" or "vertical".
group_IDs	A character vector providing group identifiers for each experiment.
experiment_names	A character vector providing names for each experiment. The hierarchy is group > experiment, i.e. within a single group, there might be several experiments taking place (e.g. multiple extracts from the same plant species tested with plant species being the group and type of extract being the experiment).
validity_method	A character vector specifying the method for determining cell validity. It can be either "threshold" (i.e. samples are validated based on a common absorption maximum) or "samples" (i.e. samples are manually specified as invalid).
threshold	A numeric threshold value. Applied if validity_method is set to 'threshold'.
invalid_samples	A character vector containing well positions (e.g. "A-3", "B-8",...) of invalid samples. Applied if validity_method is set to 'samples'.
treatment_labels	A character vector containing treatment labels.
concentration_levels	A numeric vector containing concentration levels.
...	Additional arguments to be passed to read_plates.

Details

This function processes photometer data from multiple experiments and adds metadata based on user-set parameters if the experimental layout is repeated across plates and should by synchronized. It supports two methods for determining cell validity: "threshold" and "invalid". If "threshold" method is chosen, the validity of each cell is determined based on a specified threshold value. If "sample" method is chosen, samples at specified well positions on the plate are considered invalid. The function generates lists of treatments and concentration levels based on the direction parameter, i.e. the direction of the treatments and concentration levels applied (either horizontally or vertically on the plate). If the plate layout and, thus, the metadata changes across plates, then function tidy_plates_via_prompts might be a better choice since it helps the user to add metadata for each plate separately based on user prompts. If there is only one plate, where metadata should be added, then tidy_single_plate should be used.

For all three functions, tidy_single_plate, tidy_plates_via_params, and tidy_plates_via_prompts, to work properly, file names should provide a file identifier (i.e. "bma" in case there are additional but not relevant files in the folder), a group identifier (i.e. starting with "grp" followed by an incrementing number), an identifier for experiments (starting with "exp" followed by a number, e.g. "exp1") and an identifier for timepoints (starting with the upper- or lower-case letter t followed by an incrementing number, e.g. "T0" or "t0").

Value

A tidy tibble containing combined data and metadata from all input plates.

See Also

tidy_single_plate, tidy_plates_via_prompts

Examples

```
# Load example data
data(bma)
# Add metadata from user parameters
bma_tidy <- tidy_plates_via_params(input_data = bma,
                                   direction = "horizontal",
                                   group_IDs = paste0("Group_", letters[1:2]),
                                   experiment_names = c("Experiment 1", "Experiment 2"),
                                   validity_method = "threshold",
                                   threshold = 1,
                                   treatment_labels = LETTERS[1:8],
                                 concentration_levels = seq(from=80, to=10, length.out=8))
bma_tidy # View tidy data
```

---

tidy_plates_via_prompts

Read raw photometry data and add meta data based on user input

---

Description

Most old photometer devices save the data in plain text files. If there was ever analysis software, this is often no longer available due to increasing technical requirements or proprietary software should generally be avoided. Especially for broth microdilution assays, it is necessary to measure the photometer plates at several points in time, which means that the same samples are represented in several files with their corresponding values. Usually the data is then merged manually, which can lead to mistakes and takes up unnecessary time. In this case, the ‘tidy_plates()‘ function provides a convenient way to read in the raw files and, based on user input, add metadata on the validity of the samples, as well as treatment groups and concentration levels.

Usage

```
tidy_plates_via_prompts(
  input_data,
  direction = c("horizontal", "vertical"),
  ...
)
```

Arguments

input_data	The folder path to the files containing the raw photometer data. Data files should be given as plain text files and with timepoint identifiers in their file names (e.g. "file_T0.txt" or "file_t0.txt").
direction	A character vector specifying the orientation of the plate layout. It can be either "horizontal" or "vertical".
...	Additional arguments to be passed to read_plates.

Value

A tidy data frame containing absorption values and meta data (validity of samples as well as treatment and concentration level information).

See Also

tidy_single_plate, tidy_plates_via_params

`tidy_single_plate` *Tidy single 96-well plate via parameters*

Description

This function processes a single raw 96-well plate data from a photometer measurement by adding metadata via user-specified parameter values.

Usage

```
tidy_single_plate(
  input_data,
  direction = c("horizontal", "vertical"),
  group_ID = NULL,
  experiment_name = NULL,
  validity_method = c("threshold", "invalid"),
  threshold = NULL,
  invalid_samples = NULL,
  treatment_labels,
  concentration_levels,
  ...
)
```

Arguments

`input_data`	Either a file path or a data frames with 8 rows and 12 columns.
`direction`	A character vector specifying the orientation of the plate layout. It can be either "horizontal" or "vertical".
`group_ID`	A character vector providing group identifiers for each experiment.
`experiment_name`	A string providing the name of the experiment. The hierarchy is group > experiment, i.e. within a single group, there might be several experiments taking place (e.g. multiple extracts from the same plant species tested with plant species being the group and type of extract being the experiment).
`validity_method`	A character vector specifying the method for determining cell validity. It can be either "threshold" (i.e. samples are validated based on a common absorption maximum) or "samples" (i.e. samples are manually specified as invalid).

`threshold` A numeric threshold value. Applied if `validity_method` is set to 'threshold'.

`invalid_samples`
A character vector containing well positions (e.g. "A-3", "B-8",...) of invalid samples. Applied if `validity_method` is set to 'samples'.

`treatment_labels`
A character vector containing treatment labels.

`concentration_levels`
A numeric vector containing concentration levels.

`...` Additional arguments to be passed to `read_plates`.

Details

This function processes photometer data from a single measurement and adds metadata based on user-set parameters. It supports two methods for determining cell validity: "threshold" and "invalid". If "threshold" method is chosen, the validity of each cell is determined based on a specified threshold value. If "sample" method is chosen, samples at specified well positions on the plate are considered invalid. The function generates lists of treatments and concentration levels based on the direction parameter, i.e. the direction of the treatments and concentration levels applied (either horizontally or vertically on the plate). To add metadata to several plates at the same time, the functions `tidy_plates_via_params` and `tidy_plates_via_prompts` are recommended.

For all three functions, `tidy_plate`, `tidy_plates_via_params`, and `tidy_plates_via_prompts`, to work properly, file names should provide a file identifier (i.e. "bma" in case there are additional but unused files in the folder), an identifier for experiments (starting with "exp" followed by a number, e.g. "exp1") and an identifier for timepoints (starting with the upper- or lower-case letter t followed by a number, e.g. "T0" or "t0").

Value

A tidy tibble containing data and metadata.

See Also

`tidy_plates_via_params`, `tidy_plates_via_prompts`

Examples

```
# Load example data
data(bma)
# Add metadata from user parameters
bma_tidy <- tidy_single_plate(input_data = bma[1],
                      direction = "horizontal",
                      group_ID = "Group A",
                      experiment_name = "Experiment 1",
                      validity_method = "threshold",
                      threshold = 1,
                      treatment_labels = LETTERS[1:8],
                      concentration_levels = seq(from=80, to=10, length.out=8))
bma_tidy # View tidy data
```

validate_cells *Check validity of each cell in data frame.*

Description

validate_cells checks if samples are valid based on either a user-set threshold (i.e. a maximum absorption value) or a list of invalid samples provided by the user.

apply_validation_method evaluates whether a sample meets a user-set validity criteria based on a specified validity method.

ask_validity_method applies a user prompt to check for the validation method to apply on the samples. This can be either 'threshold' (then a maximum absorption value is asked via a call to function ask_threshold) or 'samples'

ask_threshold applies a user prompt to check for a valid absorption maximum used as a threshold.

ask_invalid_samples applies a user prompt to check for invalid samples.

update_validity updates the Validity column in a dataframe based on a specified position and combinations of factors. It sets the Validity to "invalid" for rows where the Position matches the specified position and where the combinations of factors A, B, and C match the provided group levels.

Usage

```
validate_cells(
  raw_data,
  row_names,
  col_names,
  validity_method = c("threshold", "samples"),
  threshold = NULL,
  invalid_samples = NULL
)

apply_validation_method(
  value,
  i,
  j,
  row_names,
  col_names,
  validity_method = c("threshold", "samples"),
  threshold = NULL,
  invalid_samples = NULL
)

ask_validity_method()

ask_threshold()
```

```
ask_invalid_samples()

update_validity(
  input_data,
  wp_var = "Position",
  well_positions,
  group_levels = NULL
)
```

Arguments

raw_data	The original data frame.
row_names	Names or identifiers of rows in the matrix or data frame.
col_names	Names or identifiers of columns in the matrix or data frame.
validity_method	The method used to determine validity. Either 'threshold' or 'samples'.
threshold	A threshold value used for determining validity. Only applied if 'validity_method is set to 'threshold'.
invalid_samples	A container for storing invalid samples or their indices. Only applied if 'validity_method is set to 'samples'.
value	The value to be checked for validity.
i	The row index of the value in the matrix or data frame.
j	The column index of the value in the matrix or data frame.
input_data	A dataframe containing the data to be updated.
wp_var	A character string specifying the column providing the well positions. Defaults to "Position".
well_positions	The well positions to filter the data on.
group_levels	A list specifying the combinations of factors A, B, and C to match. Each element of the list should be a vector of factor levels.

Value

validate_cells returns a data frame with validity information

apply_validation_method returns logical value indicating whether the value meets the validity criteria.

The user's validity method preference

ask_threshold returns the user-specified threshold

ask_invalid_samples returns a vector of invalid samples

update_validity returns the updated dataframe with Validity modified accordingly.

See Also

generate_experiment_list, ask_experiment_list, generate_group_list, ask_group_list, add_treatment, add_concentration, validate_cells

Examples

```
df <- data.frame(Position = c("pos1", "pos2", "pos2", "pos4", "pos4"),
                 Value = c(1, 2, 3, 4, 5),
                 Validity = c("valid", "valid", "valid", "valid", "valid"),
                 A = c("a1", "a2", "a3", "a1", "a2"),
                 B = c("b1", "b2", "b3", "b1", "b2"),
                 C = c("c1", "c2", "c3", "c1", "c2"))
updated_df <- update_validity(df,
                              well_positions = "pos2",
                              group_levels = list(A = c("a2", "a3"), B = c("b2", "b3")))
updated_df
```

Index