Package: wellr (via r-universe)

August 20, 2024

Title Utilities for Working With Plate Based Data

Version 0.3.4

Description Provides a range of utility functions for working with plate-based data from microtitre plates such as enzyme assays, bacterial growth assays and protein binding experiments. Working with column IDs, row numbers and letters and transforming between a matrix and a data frame are all made simpler.

License MIT + file LICENSE

Suggests covr, devtools, rmarkdown, roxygen2, testthat (>= 3.0.0), usethis

Config/testthat/edition 3

Config/testthat/parallel true

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.1

Depends R (>= 3.6.0)

Imports stringr, tibble, rlang, tidyr, dplyr, ggplot2, scales, plater, janitor, purrr, readr, readxl, lifecycle, cli

URL https://github.com/bradyajohnston/wellr

BugReports https://github.com/bradyajohnston/wellr/issues

Repository https://bradyajohnston.r-universe.dev

RemoteUrl https://github.com/bradyajohnston/wellr

RemoteRef HEAD

RemoteSha 3fa4a8d1defa97c87850da7a10a43a055140d9d1

Contents

`is_well_id` *Logical test for well ID format.*

Description

Logical test for well ID format.

Usage

```
is_well_id(x)
```

Arguments

`x`	A string vector.

Value

A logical vector.

Examples

```
is_well_id(c("a12", "a2", "a02", "foo1"))
```

`plate_add_meta` *Add* {plater{ *Formatted Metadata to a Dataset*

Description

Add {plater{ Formatted Metadata to a Dataset

Usage

```
plate_add_meta(data, file)
```

Arguments

`data`	A tibble::tibble() which contains a 'well' column, that will have the metadata added to it.
`file`	File patch to a {plater} formatted .csv file which contains the metadat information.

Value

a tibble::tibble()

Examples

```
file_data <- system.file(
  "extdata",
  "20220929_1steptimer20.csv",
  package = "wellr"
)
file_meta <- system.file(
  "extdata",
  "20220929_1steptimer20_metainfo.csv",
  package = "wellr"
)

plate_read_biotek(file_data) |>
  plate_add_meta(file_meta)
```

plate_read_biotek *Read Biotek Output CSV Files*

Description

Read Biotek Output CSV Files

Usage

```
plate_read_biotek(file, time_average = TRUE, second_wl = FALSE)
```

Arguments

file	The filepath to a .csv file, exported from a Biotek plate reader.
time_average	Logical, whether to average the time points across different observations (LUM / OD) and pivot them to their own columns.
second_wl	Whether to include the second wavelength when reading in data blocks, or to keep just the label and the first wavelength.

Value

tibble::tibble()

Examples

```
file_data <- system.file(
  "extdata",
  "2024-02-29_vio_GFP_main.csv",
  package = "wellr"
)

plate_read_biotek(file_data)
plate_read_biotek(file_data, second_wl = TRUE)
```

plate_read_biotek_wl *Read the* wavelength *data blocks from a* biotek .csv *file.*

Description

Read the wavelength data blocks from a *biotek* .csv file.

Usage

```
plate_read_biotek_wl(file, format_well = TRUE)
```

Arguments

file	File path to the .csv file.
format_well	Whether to format the well column of the returned dataframes.

Value

a tibble

Examples

```
file_data <- system.file(
  "extdata",
  "2024-02-29_vio_GFP_main.csv",
  package = "wellr"
)

plate_read_biotek_wl(file_data)
plate_read_biotek_wl(file_data, format = FALSE)
```

---

plate_read_meta *Read a* {plater} *Formatted Metadata File*

Description

Read a {plater} Formatted Metadata File

Usage

```
plate_read_meta(file, sep = ",")
```

Arguments

file	File path to a {plater} formatted .csv file which contains metadata information.
sep	The separator for the .csv file, defaults to ",".

Value

a tibble::tibble()

Examples

```
file_meta <- system.file(
  "extdata",
  "20220929_1steptimer20_metainfo.csv",
  package = "wellr"
)

plate_read_meta(file_meta)
```

plate_read_tecan *Read the output of Tecan Plate Readers*

Description

Read the output of Tecan Plate Readers

Usage

```
plate_read_tecan(file, temp = FALSE)
```

Arguments

file	File path to the .xlsx or .csv file.
temp	Logical, whether to include the temperature column.

Value

a tibble::tibble() of the values from the file.

Examples

```
# read a .csv tecan file
fl <- system.file(
  'extdata', 'tecanON1.csv',
  package = 'wellr'
)

plate_read_tecan(fl)

# read a .xlsx tecan file
fl <- system.file(
  'extdata', 'tecanON1.xlsx',
  package = 'wellr'
)

plate_read_tecan(fl)
```

read_meta *Read a {plater} formatted metadata file*

Description

Read a {plater} formatted metadata file

Usage

```
read_meta(file)
```

Arguments

file	to read metadata from.

Value

a tibble

Examples

```
file <- system.file("extdata", "plate_layout_96.csv", package = "wellr")
read_meta(file)
```

`well_check` *Checks if a well ID contains a letter and a number in the correct format.*

Description

Checks if a well ID contains a letter and a number in the correct format.

Usage

```
well_check(x)
```

Arguments

x	well ID to check for proper structure.

Value

A vector the same length as x with a TRUE of FALSE for a a well that can be read by wellr

`well_df_to_mat` *Convert a data.frame to a Matrix in Plate Format*

Description

Extracts row and column information from the column specified by `well_col` (defaults to 'well') and uses this to create a matrix that contains values from the `values_from` column.

Usage

```
well_df_to_mat(df, values_from, well_col = "well", plate = NULL)
```

Arguments

`df`	A dataframe containing at least a column called "well" that contains well IDs.
`values_from`	Column that contains values to populate matrix.
`well_col`	Column containing the well IDs to extract row and column information from.
`plate`	Size of the plate, to override the auto-detected plate size.

Value

A matrix containing the values that were present in the data frame in the `values_from` column

Examples

```
library(wellr)

# pivot a long data frame into a matrix based on the row and col columns,
# with values from a given column
plate <- well_plate(8, 12)
plate$value <- seq(nrow(plate))
well_df_to_mat(plate, values_from = "value")

# the function also handles missing values from a plate, filling with with NA
# if certain wells are missing
plate <- well_plate(7, 11)
plate$value <- seq(nrow(plate))
well_df_to_mat(plate, values_from = "value", plate = 384)
```

`well_df_to_mat_frames` *Convert a DataFrame to a Multi-Frame Matrix*

Description

Converts a DataFrame to a matrix where each row is a single time point, and each column is a single well. The wells are in index order, indexing across the rows so the first values are all from row A.

Usage

```
well_df_to_mat_frames(data, value, frame, well)
```

Arguments

`data`	Data frame or tibble to convert to a multi-frame matrix.
`value`	Column with value data that will be used to populate the matrix.
`frame`	Column with the frame or time data that will be used to create the rows of the matrix.
`well`	Column with the well IDs which will be used to create the columns of the matrix.

Value

a matrix where each column is a well and each row is a time point.

Examples

```
df_list <- lapply(1:10, function(x) {
  df <- wellr::well_plate()
  df$value <- rnorm(96)
  df$frame <- x
  df
})

df_frames <- do.call(rbind, df_list)

# convert the data frame to multi-frame matrix
mat <- well_df_to_mat_frames(
  data = df_frames,
  value = value,
  frame = frame,
  well = well
)

head(mat[, 1:14])
```

well_dis *Relative Distance Between Given Well and a Reference Row*

Description

Relative Distance Between Given Well and a Reference Row

Usage

```
well_dis(row, col, ref_row, ref_col)
```

Arguments

row	Numeric vector of row numbers. Must be equal in length to col.
col	Numeric vector of col numbers. Must be equal in length to row.
ref_row	Single reference row number.
ref_col	Single reference column number.

Value

a numeric vector.

Examples

```
well_dis(3, 4, ref_row = 5, ref_col = 5)
well_dis(1:8, 1:8, ref_row = well_to_row_num("C4"), ref_col = well_to_col_num("C4"))
```

well_dis_plate *Relative Distance for an Entire Plate*

Description

Relative Distance for an Entire Plate

Usage

```
well_dis_plate(ref_row, ref_col, plate = 96)
```

Arguments

ref_row	Row number to calculate distance from.
ref_col	Column number to calculate distance from.
plate	Plate size to calculate distances for.

Value

a matrix of relative distances.

Examples

```
well_dis_plate(5, 5)

image(well_dis_plate(5, 5))
```

`well_format` *Format a Well to Uppercase and Padded Numbers*

Description

Format a Well to Uppercase and Padded Numbers

Usage

```
well_format(x, num_width = 2, uppercase = TRUE)
```

Arguments

`x`	Vector of well IDs to be formatted.
`num_width`	Width to pad out number component with 0's.
`uppercase`	Logical, whether the letter should be upercase.

Value

Vector of strings as formatted well IDs.

Examples

```
well_format(c("A9", "c3", "h12"))
```

well_from_index *Convert Numeric Index to Well ID.*

Description

Convert Numeric Index to Well ID.

Usage

```
well_from_index(x, plate = 96, num_width = 2, colwise = FALSE)
```

Arguments

x	numeric index to convert to a well ID
plate	number of wells in the plate. One of c(6, 12, 24, 96, 384)
num_width	number of zeros to pad the column number with to the left.
colwise	if TRUE, index instead down the columns, so H01 is index 8, A12 is index 89 and B01 is index 2 for a 96 well plate.

Value

a column ID as a vector of strings.

Examples

```
# indexing first along the rows
well_from_index(1:20)

# indexing first down the columns
well_from_index(1:20, colwise = TRUE)
```

well_join *Join a row and column into a well ID.*

Description

Joins a column number and a row letter or number into a well ID. i.e. joins "A" and "1" to "A01" and joins "3 and "10" to "C10".

Usage

```
well_join(row, col, num_width = 2)
```

Arguments

row	either a row letter or number for to be coerced into a letter.
col	a column number.
num_width	the number of digits to pad out the column number with 0.

Value

a vector of well IDs as strings.

Examples

```
well_join(1:3, 4)
well_join(c("A", "B", "H"), 9)
well_join("C", 1:10)
```

---

well_mat_frames_to_df *Turn a Multi-Frame Matrix into a DataFrame*

Description

Takes a matrix where each row is a time point and each column is a well and converts it to a DataFrame containing a column for the well, a column for the row, a column for the column, a column for the time and a column for the values.

Usage

```
well_mat_frames_to_df(mat, value = "value")
```

Arguments

mat	A matrix where each column is a well and each row is a frame from a multi time point experiment.
value	Name for the column containing the values in the resulting data frame.

Value

a tibble

Examples

```
df_list <- lapply(1:10, function(x) {
  df <- wellr::well_plate()
  df$value <- rnorm(96)
  df$frame <- x
  df
})

df_frames <- do.call(rbind, df_list)
```

```
# convert the data frame to multi-frame matrix
mat <- well_df_to_mat_frames(
  data = df_frames,
  value = value,
  frame = frame,
  well = well
)

head(mat[, 1:14])

# convert the matrix back to a dataframe
well_mat_frames_to_df(mat)
```

`well_mat_to_df` *Convert a Plate Matrix to a DataFrame*

Description

Takes a matrix that is formatted as a plate, and converts it to a long-format 'tidy' data frame with a column for the well ID, the row, the column and the values that were in the matrix.

Usage

```
well_mat_to_df(matrix, value_col = "value")
```

Arguments

`matrix`	A matrix object to be converted.
`value_col`	The name of the value column in the final DataFrame.

Value

a tibble

Examples

```
mat <- matrix(rnorm(96), ncol = 12)
well_mat_to_df(mat, "random_values")
```

well_plate *Create a Tibble of Plate Information*

Description

Generates a tibble that contains row, col, and well ID for the size of the plate specified in nrow and ncol. If vectors of length > 1 are supplied to either nrow or ncol, the contents of the vectors are used instead of their numeric value.

Usage

```
well_plate(nrow = 8, ncol = 12)
```

Arguments

nrow	Number of rows to have in the generated plate.
ncol	NUmber of columns to have in the generated plate.

Value

a tibble

Examples

```
well_plate(nrow = 8, ncol = 12)
well_plate(nrow = 4, ncol = 6)
```

well_plot *Plot a Plate Map*

Description

Plot a Plate Map

Usage

```
well_plot(data, well, value, colour = "black")
```

Arguments

data	Dataframe containing data.
well	Name of the column with well IDs.
value	Name of the column with the values to plot.
colour	Colour of the well borders.

Value

ggplot2 object.

Examples

```
dat <- wellr::well_plate()[, "well"]
dat$value <- rnorm(96)
well_plot(dat, well, value)
```

`well_reorder_df` *Reorder a Plate-Based DataFrame*

Description

Reorder a Plate-Based DataFrame

Usage

```
well_reorder_df(data, well_col = "well", row_col = NULL, col_col = NULL)
```

Arguments

`data`	Data frame to reorder, than contains a row column and a col column.
`well_col`	Column containing the well IDs.
`row_col`	Column containing the row letters or numbers.
`col_col`	Column containing the column numbers.

Value

A data.frame that has been reordered according to the `row` & `col` columns.

Examples

```
df <- well_plate(nrow = 8, ncol = 12)
df <- df[sample(1:96, 96), ]
head(df)

df <- well_reorder_df(df)
head(df)
```

well_to_col_num *Extract number from well ID.*

Description

Extract number from well ID.

Usage

```
well_to_col_num(x)
```

Arguments

x	Well ID as string in format "A10"

Value

Numeric verctor of column numbers extracted from well ID.

Examples

```
well_to_col_num(c("A10", "c3", "h12"))
well_to_col_num("H08")
well_to_col_num("h8")
```

well_to_index *Converts Well ID to a Numeric Index*

Description

Indexes along rows first, so A12 is index 12 and B01 is index 13 for a 96 well plate.

Usage

```
well_to_index(x, plate = 96, colwise = FALSE)
```

Arguments

x	string well ID
plate	size of the plate. One of c(6, 12, 24, 96, 384)
colwise	if TRUE, index instead down the columns, so H01 is index 8, A12 is index 89 and B01 is index 2 for a 96 well plate.

Value

numeric well index.

Examples

```
# indexing along the row first
well_to_index(c("A10", "c3", "h12"))
well_to_index("H08")
well_to_index("h8")
well_to_index("C20")

# indexing instead down the column first
well_to_index(c("A10", "c3", "h12"), colwise = TRUE)
well_to_index("H08", colwise = TRUE)
well_to_index("h8", colwise = TRUE)
well_to_index("C20", colwise = TRUE)
```

---

well_to_row_let *Extracts letter from well ID.*

Description

Extracts letter from well ID.

Usage

```
well_to_row_let(x)
```

Arguments

x	Well ID as string.

Value

a string which is the letter extracted from a well ID.

Examples

```
well_to_row_let(c("A10", "c3", "h12"))
well_to_row_let("H08")
well_to_row_let("h8")
well_to_row_let("C20")
```

well_to_row_num *Convert Well ID to Row Number*

Description

Calculates the row, but returns it as a number.

Usage

```
well_to_row_num(x)
```

Arguments

x	Well ID as a string.

Value

Numeric row number.

Examples

```
well_to_row_num(c("A10", "c3", "h12"))
well_to_row_num("H08")
well_to_row_num("h8")
well_to_row_num("C20")
```

Index