Package 'plater'

June 27, 2017

Title Read, Tidy, and Display Data from Microtiter Plates

Version 1.0.1

Description Tools for interacting with data from experiments done in microtiter plates. Easily read in plate-shaped data and convert it to tidy format, combine plate-shaped data with tidy data, and view tidy data in plate shape.

Depends R (>= 3.1.0)

Imports utils, dplyr (> 0.4.3)

Suggests testthat, knitr, rmarkdown

License GPL-3

LazyData true

URL https://github.com/ropenscilabs/plater

BugReports https://github.com/ropenscilabs/plater/issues

VignetteBuilder knitr

RoxygenNote 6.0.1

NeedsCompilation no

Author Sean Hughes [aut, cre]

Maintainer Sean Hughes <smhughes@uw.edu>

Repository CRAN

Date/Publication 2017-06-26 22:16:11 UTC

R topics documented:

add_plate *Read a plater-formatted file and combine it with an existing data frame.*

Description

Converts data from `plater` format to a data frame with one well per row and merges it into an existing data frame by well name.

Usage

```
add_plate(data, file, well_ids_column)
```

Arguments

`data`	The data frame to merge the file into. Must contain a column with well names.
`file`	The path of a .csv file formatted as described in `read_plate`.
`well_ids_column`	The name of the column in `data` containing the well IDs.

Details

If data contains more wells than in `file`, NA will be added to the merged column for those wells. If the file contains more wells than `data`, an error will result.

Value

Returns data with as many new columns as plates in `file`. Empty wells are indicated with NA.

Examples

```
# Part of the data is tidy
file <- system.file("extdata", "example-2-part-A.csv", package = "plater")
data <- read.csv(file)

# Part of the data is plate-shaped
plate_shaped <- system.file("extdata", "example-2-part-B.csv", package = "plater")

# Combine the two
data <- add_plate(
   data = data,
   file = plate_shaped,
   well_ids_column = "Wells")

# Now data are tidy
head(data)
```

check_plater_format *Check whether a file is in plater format.*

Description

Runs the provided file through a number of diagnostics to determine whether it is a valid plater format file and displays information about any deficiencies found.

Usage

```
check_plater_format(file)
```

Arguments

file	The path of the file to check

Value

Displays a number of messages as it checks the file. Will stop with a descriptive error message if the file is not formatted correctly.

Examples

```
file_path <- system.file("extdata", "example-1.csv", package = "plater")

data <- check_plater_format(file_path)
```

plater *Tools to Make it Easy to Work with Microtiter Plate-Shaped Data*

Description

plater defines a simple, plate-shaped file format for data storage, so it's easy to remember the experimental design. The package provides functions to seamlessly convert between that format and a tidy data frame that's optimal for analysis. check_plater_format is provided to help you manage plate-shaped files.

You can work with purely plate-shaped data (read_plate and read_plates), as well as with a combination of plate-shaped data and tidy data (add_plate). It further allows easy plate-shaped visualization of tidy data (view_plate).

`read_plate` *Read a plater-formatted file and turn it into a tidy data frame.*

Description

Converts data from `plater` format to a data frame with one well per row identified by well name.

Usage

```
read_plate(file, well_ids_column = "Wells")
```

Arguments

`file`	The path of a .csv file formatted as described below.
`well_ids_column`	The name to give the column that will contain the well identifiers. Default "Wells".

Value

Returns a data frame with each well as a row. One column will be named with `well_ids_column` and contain the well names (A01, A02..). There will be as many additional columns as layouts in `file`. Empty wells are omitted.

`plater` format

The .csv file should be formatted as a microtiter plate. The top-left most cell contains the name to use for the column representing that plate. For example, for a 96-well plate, the subsequent wells in the top row should be labeled 1-12. The subsequent cells in the first column should be labeled A-H. That is:

ColName	**1**	**2**	**3**	**...**
A	A01	A02	A03	...
B	B01	B02	B03	...
...	...	...	...	...

In this example, the cells within the plate contain the well IDs ("A01", "A02"), but they may contain arbitrary characters: numbers, letters, or punctuation. Any cell may also be blank.

Note that Microsoft Excel will sometimes include cells that appear to be blank in the .csv files it produces, so the files may have spurious columns or rows outside of the plate, causing errors. To solve this problem, copy and paste just the cells within the plate to a fresh worksheet and save it.

Multiple columns

Multiple columns of information about a plate can be included in a single file. After the first plate, leave one row blank, and then add another plate formatted as described above. (The "blank" row

should appear as blank in a spreadsheet editor, but as a row of commas when viewed as plain text.) As many plates as necessary can be included in a single file (e.g. data measured, subject, treatment, replicate, etc.).

Examples

```
file_path <- system.file("extdata", "example-1.csv", package = "plater")

# Data are stored in plate-shaped form
data <- read_plate(
   file = file_path,
   well_ids_column = "Wells")

# Now data are tidy
head(data)
```

`read_plates` *Read multiple plater-formatted files and combine result into one data frame.*

Description

A wrapper around `read_plate` that handles multiple plates and combines them all into a single data frame.

Usage

```
read_plates(files, plate_names = NULL, well_ids_column = "Wells")
```

Arguments

`files`	A character vector with the paths of one or more plater-formatted .csv files.
`plate_names`	A character vector the same length as `files` with the names to give the individual plates in the resulting data frame. Defaults to the file names (stripped of path and .csv).
`well_ids_column`	The name to give the column that will contain the well identifiers. Default "Wells".

Value

Returns a data frame like that returned by `read_plate`, containing the data from all of the plates. The plates will be identified with a column called "Plate" containing the names given in `plate_names`.

Examples

```
# Combine multiple files into one tidy data frame
file1 <- system.file("extdata", "example-1.csv", package = "plater")
file2 <- system.file("extdata", "more-bacteria.csv", package = "plater")

# Data are stored in plate-shaped form
data <- read_plates(
   files = c(file1, file2),
   plate_names = c("Experiment 1", "Experiment 2"),
   well_ids_column = "Wells")

# Data from both plates are tidy and in the same data frame
head(data)
```

`view_plate` *Displays the data in the form of a microtiter plate.*

Description

Displays the data in the form of a microtiter plate.

Usage

```
view_plate(data, well_ids_column, columns_to_display, plate_size = 96)
```

Arguments

`data`	A data frame containing the data
`well_ids_column`	The name of the column in `data` containing the well IDs.
`columns_to_display`	A vector of the names of one or more columns you'd like to display.
`plate_size`	The number of wells in the plate. Must be 12, 24, 48, 96 or 384. Default 96.

Value

A depiction of the data in `columns_to_display` as though laid out on a microtiter plate with `plate_size` wells.

Examples

```
# Generate some tidy data
data <- data.frame(Wells = paste0(LETTERS[1:3], 0, rep(1:4, each = 3)),
Species = rep(c("Alien", "Human", "Cat"), 4),
OxygenProduction = round(rnorm(12), 3))
head(data)

# See which wells had cells from which species and the amount of oxygen
# produced for each well
view_plate(data, "Wells", c("Species", "OxygenProduction"), 12)
```

Index