Package 'ddpcr'

January 3, 2019

Title Analysis and Visualization of Droplet Digital PCR in R and on the Web

Version 1.11

Description An interface to explore, analyze, and visualize droplet digital PCR (ddPCR) data in R. This is the first non-proprietary software for analyzing two-channel ddPCR data. An interactive tool was also created and is available online to facilitate this analysis for anyone who is not comfortable with using R.

URL https://github.com/daattali/ddpcr

BugReports https://github.com/daattali/ddpcr/issues

Depends R (>= 3.1.0)

Imports DT (>= 0.2), dplyr (>= 0.5.0), ggplot2 (>= 2.2.0), lazyeval (>= 0.1.10), magrittr (>= 1.5), mixtools (>= 1.0.2), plyr (>= 1.8.1), readr (>= 0.1.0), shiny (>= 0.11.0), shinyjs (>= 0.4.0)

Suggests ggExtra (>= 0.3.0), graphics, grid (>= 3.2.2), gridExtra (>= 2.0.0), knitr (>= 1.7), rmarkdown, stats, testthat (>= 0.11.0), utils

License MIT + file LICENSE

LazyData true

VignetteBuilder knitr

RoxygenNote 5.0.1

NeedsCompilation no

Author Dean Attali [aut, cre]

Maintainer Dean Attali <daattali@gmail.com>

Repository CRAN

Date/Publication 2019-01-03 15:10:10 UTC

R topics documented:

analysis_complete *Is the analysis complete?*

Description

Check if a ddPCR plate has been fully analyzed or if there are remaining steps.

Usage

```
analysis_complete(plate)
```

Arguments

plate	A ddPCR plate

Value

TRUE if the plate's analysis has been fully carried out; FALSE otherwise.

See Also

status
analyze

analyze *Run analysis on a ddPCR plate*

Description

Every ddPCR plate has a set of defined steps that are taken in order, that together constitute "analyzing" the plate. Calling the analyze function will perform all the analysis steps, which may take several minutes. Running the analysis will classify the droplets in the plate into clusters (available via plate_data) and will add variables to the plate metadata (available via plate_meta).

Usage

```
analyze(plate, restart = FALSE)
```

Arguments

plate	A ddPCR plate
restart	If TRUE, then run the analysis from the beginning; othrewise, continue from the last step that was performed.

Details

This function will run an analysis to completion. If you want to run each step one at a time, use `next_step`.

Value

The analyzed ddPCR plate

Note

Most analysis steps result in some progress messages being printed to the screen. You can turn off these messages by disabling the verbose option with the command `options(ddpcr.verbose = FALSE)`.

See Also

`next_step`
`plot.ddpcr_plate`
`new_plate`
`steps`
`plate_data`
`plate_meta`

Examples

```
## Not run:
plate <- new_plate(sample_data_dir(), type = plate_types$custom_thresholds)
plate <- analyze(plate)

## End(Not run)
```

`clusters` *Potential droplet clusters for a plate type*

Description

Each ddPCR plate type has a specific set of potential clusters the droplets can be assigned to.

Usage

```
clusters(plate)
```

Arguments

`plate`	a ddPCR plate.

Details

See the README for more information on plate types.

Value

A character vector with the names of the clusters supported by the plate type.

Examples

```
## Not run:
dir <- sample_data_dir()
new_plate(dir) %>% clusters
new_plate(dir, plate_types$fam_positive_pnpp) %>% clusters

## End(Not run)
```

`custom_thresholds` *Plate type: custom thresholds*

Description

The `custom_thresholds` plate type is used when you want to gate ddPCR droplet data into four quadrants according to HEX and FAM values that you manually set. All wells in the plate will use the same threshold values.

Details

Plates with this type have only three analysis steps: `INITIALIZE`, `REMOVE_OUTLIERS`, and `CLASSIFY` (according to the custom thresholds).

Plates with this type have the following droplet clusters: `UNDEFINED`, `OUTLIER`, `EMPTY` (bottom-left quadrant), `X_POSITIVE` (bottom-right quadrant), `Y_POSITIVE` (top-left quadrant), `BOTH_POSITIVE` (top-right quadrant).

See the README for more information on plate types.

See Also

`plate_types`
`x_threshold`
`y_threshold`
`thresholds`
`analyze`
`remove_outliers`
`classify_thresholds`

Examples

```
## Not run:
plate <- new_plate(sample_data_dir(), type = plate_types$custom_thresholds)
type(plate)
plate %>% analyze %>% plot

## End(Not run)
```

ddpcr — *Analysis and visualization of Droplet Digital PCR data.*

Description

An interface to explore, analyze, and visualize droplet digital PCR (ddPCR) data in R. This is the first known non-proprietary software for analyzing ddPCR data. An interactive tool was also created and is available online to facilitate this analysis for anyone who is not comfortable with using R.

Use the web tool or read the full documentation on GitHub.

ddpcr_plate — *Plate type: ddPCR plate*

Description

The default plate type that all other plates inherit from. If you initialize a ddPCR plate without specifying a plate type, `ddpcr_plate` will be the plate's type.

Details

Plates with this type have the following analysis steps: `INITIALIZE`, `REMOVE_FAILURES`, `REMOVE_OUTLIERS`, `REMOVE_EMPTY`.

Plates with this type have the following droplet clusters: `UNDEFINED`, `FAILED`, `OUTLIER`, `EMPTY`.

See the README for more information on plate types.

See Also

`plate_types`
`remove_failures`
`remove_outliers`
`remove_empty`

Examples

```
## Not run:
plate <- new_plate(sample_data_dir(), type = plate_types$ddpcr_plate)
type(plate)
plate %>% analyze %>% plot

## End(Not run)
```

`fam_positive_pnpp` *Plate type: FAM-positive PNPP*

Description

A ddPCR plate of type `fam_positive_pnpp`, which can also be expressed as (FAM+)/(FAM+HEX+), is a subtype of both `pnpp_experiment` and `wildtype_mutant_pnpp`. Use this plate type if your data has three main clusters of droplets: double-negative (empty droplets), FAM+HEX+ (wildtype droplets) and FAM+HEX- (mutant droplets).

Details

Plates with this type have the following analysis steps: `INITIALIZE`, `REMOVE_FAILURES`, `REMOVE_OUTLIERS`, `REMOVE_EMPTY`, `CLASSIFY`, `RECLASSIFY`.

Plates with this type have the following droplet clusters: `UNDEFINED`, `FAILED`, `OUTLIER`, `EMPTY` (double-negative), `RAIN` (not empty but not wildtype nor negative), `POSITIVE` (wildtype), `NEGATIVE` (mutant).

See the README for more information on plate types.

See Also

`plate_types`
`wildtype_mutant_pnpp`
`hex_positive_pnpp`
`analyze`
`remove_failures`
`remove_outliers`
`remove_empty`
`classify_droplets`
`reclassify_droplets`

Examples

```
## Not run:
plate <- new_plate(sample_data_dir(), type = plate_types$fam_positive_pnpp)
type(plate)
plate %>% analyze %>% plot

## End(Not run)
```

hex_positive_pnpp *Plate type: HEX-positive PNPP*

Description

A ddPCR plate of type hex_positive_pnpp, which can also be expressed as (HEX+)/(FAM+HEX+), is a subtype of both pnpp_experiment and wildtype_mutant_pnpp. Use this plate type if your data has three main clusters of droplets: double-negative (empty droplets), FAM+HEX+ (wildtype droplets) and HEX+FAM- (mutant droplets).

Details

Plates with this type have the following analysis steps: INITIALIZE, REMOVE_FAILURES, REMOVE_OUTLIERS, REMOVE_EMPTY, CLASSIFY, RECLASSIFY.

Plates with this type have the following droplet clusters: UNDEFINED, FAILED, OUTLIER, EMPTY (double-negative), RAIN (not empty but not wildtype nor negative), POSITIVE (wildtype), NEGATIVE (mutant).

See the README for more information on plate types.

See Also

plate_types
wildtype_mutant_pnpp
fam_positive_pnpp
analyze
remove_failures
remove_outliers
remove_empty
classify_droplets
reclassify_droplets

Examples

```
## Not run:
plate <- new_plate(sample_data_dir(), type = plate_types$hex_positive_pnpp)
type(plate)
plate %>% analyze %>% plot

## End(Not run)
```

launch *Run the interactive analysis tool (Shiny app) in a web browser*

Description

In addition to the functions provided in this package, the ddpcr package also provides an interactive tool that can be used to analyze ddPCR data more easily. The tool will be launched in a web browser.

Usage

```
launch()
```

load_plate *Load a previously saved ddPCR plate*

Description

Reloads a plate that has been saved with save_plate.

Usage

```
load_plate(file)
```

Arguments

file	Name of the file where the plate was saved.

Value

The plate that was saved in the given file.

See Also

save_plate

Examples

```
plate <- new_plate(sample_data_dir())
save_plate(plate, "myplate")
plate2 <- load_plate("myplate")
plate3 <- load_plate("myplate.rds")
identical(plate, plate2)
identical(plate, plate3)
unlink("myplate.rds")
```

name *Plate name*

Description

Get or set the name of a dataset.

Usage

```
name(plate)

name(plate) <- value
```

Arguments

plate	A ddpcrPlate
value	New name

Value

Plate name

Examples

```
## Not run:
plate <- new_plate(sample_data_dir())
name(plate)
name(plate) <- "foo"
name(plate)

## End(Not run)
```

new_plate *Create a new ddPCR plate*

Description

Any ddPCR analysis must start by creating a ddPCR plate object. Use this function to read ddPCR data into R and create a plate object that can then be analyzed.

Usage

```
new_plate(dir, type, data_files, meta_file, name, params)
```

Arguments

dir	The directory containing the ddPCR droplet data files, and potentially the plate results file
type	A ddPCR plate type (see plate_types)
data_files	If dir is not provided, you can provide a vector of file paths to the ddPCR droplet data files.
meta_file	If dir is not provided, you can provide a file path to the ddPCR results file.
name	Name of the dataset. If not provided, the name will be guessed based on the filenames.
params	List of parameters to set for the plate. Only advanced users should consider using this feature.

Details

See the README for more information on plate types.

Value

A new ddPCR plate object with droplet data loaded that is ready to be analyzed.

Providing ddPCR data

The first step to using the ddpcr package is to get the ddPCR data into R. This package uses as input the data files that are exported by QuantaSoft. For a dataset with 20 wells, QuantaSoft will create 20 well files (each ending with "_Amplitude.csv") and one results file. The well files are essential for analysis since they contain the actual droplet data, and the results file is optional because the only information used from it is the mapping from well IDs to sample names.

The easiest way to use your ddPCR data with this package is to Export the data from QuantaSoft into some directory, and providing that directory as the dir argument. This way, this package will automatically find all the data files as well as the results file. Alternatively, you can provide the data files (well files) manually as a list of filenames using the data_files argument. If you use the data_files argument instead of dir, you can also optionally provide the results file as the meta_file argument. If no results file is provided then the wells will not be mapped to their sample names.

Plate parameters

Every plate has a set of default parameters that are used in the analysis. You can see all the parameters of a plate with the params function. If you want to provide different values for some parameters when initializing a plate, you can do that with the params argument. This is considered an advanced feature.

For example, if you inspect the parameters of any ddPCR plate, you will see that by defalt the random seed used by default is 8. If you want to create a new plate that uses a different random seed, you could do so like this:

```
plate <- new_plate(sample_data_dir(), params = list('GENERAL' = list('RANDOM_SEED' = 10)))
plate
```

Most numeric parameters that are used in the algorithms of the analysis steps can be modified in a similar fashion. This can be used to fine-tune the analysis of a plate if you require different parameters.

See Also

plate_types
type
reset
analyze
plot.ddpcr_plate
params

Examples

```
## Not run:
plate <- new_plate(sample_data_dir())

## End(Not run)
```

---

next_step *Run the next step in an analysis*

Description

Every ddPCR plate has a set of defined steps that are taken in order, that together constitute "analyzing" the plate. Calling the `next_step` function will run the next step in the analysis, which may take several minutes. If you want to run all the remaining steps at once, use `analyze` instead.

Usage

```
next_step(plate, n = 1)
```

Arguments

`plate`	A ddPCR plate
`n`	The number of steps to run

Value

The ddPCR plate after running the next step

See Also

plot.ddpcr_plate
analyze
steps
plate_data
plate_meta

Examples

```
## Not run:
plate <- new_plate(sample_data_dir(), type = plate_types$custom_thresholds)
plate <- next_step(plate)

## End(Not run)
```

`params` *Plate parameters*

Description

Every ddPCR plate object has adjustable parameters associated with it. Each parameter belongs to a category of parameters, and has a unique name. For example, there are general parameters (category 'GENERAL') that apply to the plate as a whole, and each analysis step has its own set of parameters that are used for the algorithm in that step.

You can either view all parameters of a plate by not providing any arguments, view all parameters in a category by providing the category, or view the value of a specific parameter by providing both the category and the parameter name.

Usage

```
params(plate, category, name)

params(plate, category, name) <- value
```

Arguments

`plate`	A ddPCR plate
`category`	Category of parameters
`name`	Parameter name
`value`	New parameter value

Details

Setting new parameter values should only be done by advanced users. Note that if you change any parameters, you need to re-run the analysis in order for the parameter changes to take effect.

Tip: it can be easier to visually inspect the parameters by wrapping the return value in a `str()`.

Warning: Do not directly set the GENERAL-X_VAR or GENERAL-Y_VAR parameters. Instead, use `x_var` or `y_var`.

Value

If no category is provided, return all parameters. If a category is provided, return all parameters in that category. If both a category and a name are provided, return the value of the specific parameter.

See Also

x_var

Examples

```
## Not run:
plate <- new_plate(sample_data_dir())

# retrieving plate parameters
str(params(plate))
str(params(plate, 'GENERAL'))
params(plate, 'GENERAL', 'RANDOM_SEED')

# setting plate parameters
params(plate, 'GENERAL', 'RANDOM_SEED') <- 10
str(params(plate, 'GENERAL'))

## End(Not run)
```

---

plate_data *Plate data (droplets data)*

Description

The main piece of information in every ddPCR plate is the droplets data, which contains the fluorescence intensities for every single droplet in every well. After a ddPCR plate gets analyzed, this data also includes the assigned cluster for each droplet. The plate data may be useful programatically, but it's not very useful to a human, so if you want to visualize the plate data you should instead plot it using plot.ddpcr_plate.

Usage

```
plate_data(plate)
```

Arguments

plate	A ddPCR plate

Value

A dataframe containing all the droplets in the plate, along with the assigned cluster of each droplet.

See Also

plate_meta
plot.ddpcr_plate

Examples

```
## Not run:
plate <- new_plate(sample_data_dir())
plate_data(plate)

## End(Not run)
```

---

plate_meta *Plate metadata*

Description

The metadata is a collection of variables that describe each well in the plate. The metadata of an unanalyzed plate only contains basic information about each well, such as the sample name, whether the well was used, and the number of droplets in the well. Analyzing a plate adds many more variables to the metadata, such as the number of empty droplets, the number of outliers, the template concentration, and more.

Usage

```
plate_meta(plate, only_used = FALSE)
```

Arguments

plate	A ddPCR plate
only_used	If TRUE, only return metadata for wells that are used in this plate (wells that have any data)

Value

A dataframe containing the plate metadata

See Also

plate_data plot.ddpcr_plate

Examples

```
## Not run:
plate <- new_plate(sample_data_dir())
plate %>% plate_meta(only_used = TRUE)
plate %>% analyze %>% plate_meta(only_used = TRUE)

## End(Not run)
```

plate_types *Supported plate types*

Description

Each ddPCR plate has a plate type which determines what type of analysis to run on the data. `plate_types` is a list containing the plate types that are supported. If no plate type is specified, the default assumed type is `ddpcr_plate`.

The most useful built-in plate types are: `fam_positive_pnpp`, `hex_positive_pnpp`, `custom_thresholds`.

For full details on the differences between plate types or to learn how to add a new plate type, see the package README.

See Also

`new_plate`
`fam_positive_pnpp`
`hex_positive_pnpp`
`custom_thresholds`
`pnpp_experiment`
`wildtype_mutant_pnpp`
`ddpcr_plate`
`type`

Examples

```
## Not run:
dir <- sample_data_dir()
new_plate(dir, type = plate_types$ddpcr_plate)
new_plate(dir, type = plate_types$custom_thresholds)
new_plate(dir, type = plate_types$fam_positive_pnpp)

## End(Not run)
```

plot.custom_thresholds *Plot a ddPCR plate of type custom thresholds*

Description

Same plot as `plot.ddpcr_plate` but with a few extra features that are specific to plates with custom thresholds. Take a look at `plot.ddpcr_plate` to see all supported parameters and more information.

Usage

```
## S3 method for class 'custom_thresholds'
plot(x, wells, samples, ...,
  show_thresholds = TRUE, col_thresholds = "black",
  show_drops_empty = TRUE, col_drops_x_positive = "green3",
  col_drops_y_positive = "blue", col_drops_both_positive = "orange")
```

Arguments

x	A ddPCR plate.
wells	Only plot selected wells. Supports range notation, see subset.ddpcr_plate.
samples	Only plot selected samples.
...	Parameters to pass to plot.ddpcr_plate.
show_thresholds	If TRUE, show the thresholds.
col_thresholds	The colour of the threshold lines.
show_drops_empty	Whether or not to show the droplets defined as empty.
col_drops_x_positive	The colour to use for droplets that are in the X+Y- quadrant.
col_drops_y_positive	The colour to use for droplets that are in the X-Y+ quadrant.
col_drops_both_positive	The colour to use for droplets that are in the X+Y+ quadrant.

Value

A ggplot2 plot object.

See Also

plot.ddpcr_plate
custom_thresholds

Examples

```
## Not run:
plate <- new_plate(sample_data_dir(), type = plate_types$custom_thresholds)
plate %>% set_thresholds(c(5500, 8000)) %>% analyze %>% plot

## End(Not run)
```

`plot.ddpcr_plate` *Plot a ddPCR plate*

Description

Plot the data of a ddPCR plate. A plate can be plotted throughout any stage of the analysis, and the most up-to-date data will be shown. For example, a plot performed after initializing a plat will show all the raw data, but a plot performed after analyzing a plate will show information such as empty drops and failed wells.

Usage

```
## S3 method for class 'ddpcr_plate'
plot(x, wells, samples, superimpose = FALSE,
  show_full_plate = FALSE, show_drops = TRUE, show_drops_empty = FALSE,
  show_drops_outlier = FALSE, show_failed_wells = TRUE,
  col_drops = "black", col_drops_undefined = col_drops,
  col_drops_failed = col_drops, col_drops_empty = col_drops,
  col_drops_outlier = "orange", bg_plot = "transparent",
  bg_failed = "#111111", bg_unused = "#FFFFFF", alpha_drops = 0.2,
  alpha_drops_outlier = 1, alpha_bg_failed = 0.7, xlab = x_var(plate),
  ylab = y_var(plate), title = NULL, show_grid = FALSE,
  show_grid_labels = FALSE, drops_size = 1, text_size_title = 14,
  text_size_row_col = 12, text_size_axes_labels = 12,
  text_size_grid_labels = 12, ...)
```

Arguments

`x`	A ddPCR plate.
`wells`	Only plot selected wells. Supports range notation, see `subset.ddpcr_plate`.
`samples`	Only plot selected samples.
`superimpose`	If `TRUE`, show all wells superimposed in one plot; otherwise, show wells in a grid.
`show_full_plate`	If `TRUE`, show full 96-well plate; otherwise, show only plate rows and columns that are used.
`show_drops`	Whether or not to show the droplets. Setting to `FALSE` is not useful if the droplets are the only thing shown in the plot, but it can be useful if there is other information depicated in the plot, such as any background colours or text that may appear in each well.
`show_drops_empty`	Whether or not to show the droplets defined as empty. See 'Droplet visibility options' below.
`show_drops_outlier`	Whether or not to show the droplets defined as outliers. See 'Droplet visibility options' below.

`show_failed_wells`
: Whether or not to include wells that are deemed as failed ddPCR runs.

`col_drops`
: The default colour to use for any droplet.

`col_drops_undefined`
: The colour to use for droplets that have not been analyzed yet. See 'Droplet visibility options' below.

`col_drops_failed`
: The colour to use for droplets in failed wells. See 'Droplet visibility options' below.

`col_drops_empty`
: The colour to use for empty droplets. See 'Droplet visibility options' below.

`col_drops_outlier`
: The colour to use for outlier droplets. See 'Droplet visibility options' below.

`bg_plot`
: The background colour for the plot.

`bg_failed`
: The background colour to use for failed wells.

`bg_unused`
: The background colour to use for unused wells.

`alpha_drops`
: The transparency of droplets.

`alpha_drops_outlier`
: The transparency of outlier droplets. See 'Droplet visibility options' below.

`alpha_bg_failed`
: The transparency of the background of failed wells.

`xlab`
: The label on the X axis.

`ylab`
: The label on the Y axis.

`title`
: The title for the plot.

`show_grid`
: Whether or not to show grid lines.

`show_grid_labels`
: Whether or not to show numeric labels for the grid lines along the axes.

`drops_size`
: Size of droplets.

`text_size_title`
: Text size of the title.

`text_size_row_col`
: Text size of the row and column labels.

`text_size_axes_labels`
: Text size of the X/Y axis labels.

`text_size_grid_labels`
: Text size of the numeric grid line labels.

`...`
: Ignored.

Value

A ggplot2 plot object.

Droplet visibility options

To make it easier to support any plate type with any types of droplet clusters, there are three categories of special parameters that can always be used:

- `show_drops_*` Whether or not to show a specific group of droplets.
- `col_drops_*` What colour to use for a specific group of droplets.
- `alpha_drops_*` What transparency to use for a specific group of droplets.

The `*` in the parameter name can be replaced by the name of any droplet cluster. Use the `clusters` function to find out what clusters the droplets in a plate can be assigned to.

For example, the default clusters that exist in a plain `ddpcr_plate` are "UNDEFINED", "FAILED", "OUTLIER", and "EMPTY". This means that if you want to hide the empty drops and make the transparency of drops in failed wells 0.5, you could add the two parameters `show_drops_empty = FALSE` and `alpha_drops_failed = 0.5`. Note that letter case is not important. If another plate type defines a new clsuter of type "MUTANT" and you want to show these drops in red, you can add the parameter `col_drops_mutant = "red"`.

Note that some of the more common combinations of these parameters are defined by default (for example, `col_drops_failed` is defined in the list of parameters), but these three parameter categories will work for any cluster type.

Extending ddpcr_plate

If you create your own plate type, this default plot function might be enough if there is no extra information you want to display in a plot. If you do need to provide a more customized plot function, it can be a good idea to use the output from this plot function as a basis and only add the code that is necessary to append to the plot. See `plot.custom_thresholds` as an example of how to extend this plot function.

Examples

```
## Not run:
plate <- new_plate(sample_data_dir())
plot(plate)
plate <- plate %>% analyze
plot(plate)
plot(plate, "A01:C05", show_drops_empty = TRUE, col_drops_empty = "red")

## End(Not run)
```

---

plot.wildtype_mutant_pnpp

Plot a ddPCR plate of type wildtype/mutant PNPP

Description

Same plot as `plot.pnpp_experiment` but with a few extra features that are specific to wildtype/mutant PNPP plates. Take a look at `plot.pnpp_experiment` to see all supported parameters and more information.

Usage

```
## S3 method for class 'wildtype_mutant_pnpp'
plot(x, wells, samples, ...,
  col_drops_mutant = "purple3", col_drops_wildtype = "green3",
  col_drops_rain = "black", show_mutant_freq = TRUE,
  text_size_mutant_freq = 4, alpha_drops_low_mutant_freq = 0.5,
  show_low_high_mut_freq = TRUE, bg_mutant = "purple3",
  bg_wildtype = "green3", alpha_bg_low_high_mut_freq = 0.1)
```

Arguments

x	A ddPCR plate.
wells	Only plot selected wells. Supports range notation, see subset.ddpcr_plate.
samples	Only plot selected samples.
...	Parameters to pass to plot.pnpp_experiment.
col_drops_mutant	The colour to use for mutant droplets.
col_drops_wildtype	The colour to use for wildtype droplets.
col_drops_rain	The colour to use for rain droplets.
show_mutant_freq	If TRUE, show the mutant frequency as a percentage on each well.
text_size_mutant_freq	Text size of the printed mutant frequencies.
alpha_drops_low_mutant_freq	Transparency of mutant droplets in wells with mostly wildtype droplets. In wells where there are very few mutant droplets, it might be useful to make them more visible by increasing their transparency.
show_low_high_mut_freq	Differentiate between wells with a high vs low mutant frequency by having a different background colour to the well.
bg_mutant	The background colour for wells that have a significant mutant cluster.
bg_wildtype	The background colour for wells that have mostly wildtype drops.
alpha_bg_low_high_mut_freq	The transparency value for bg_mutant and bg_wildtype.

Value

A ggplot2 plot object.

See Also

plot.ddpcr_plate
plot.pnpp_experiment
wildtype_mutant_pnpp

Examples

```
## Not run:
plate <- new_plate(sample_data_dir(), type = plate_types$fam_positive_pnpp) %>% analyze
wells_wildtype(plate)
plot(plate)
plate <- plate %>% analyze
plot(plate)
plot(plate, "A01:C05", col_drops_rain = "blue")

## End(Not run)
```

`pnpp_experiment` *Plate type: PNPP experiment*

Description

PNPP stands for "Positive-Negative;Positive-Positive", which is a reflection of the clusters of non-empty droplets in the wells. Use this plate type when your ddPCR data has three main clusters: double-negative (FAM-HEX-; empty droplets), double-positive (FAM+HEX+; represent the "PP" in PNPP), and singly-positive (either FAM+HEX- or HEX+FAM-; represent the "NP" in PNPP).

Details

Every `pnpp_experiment` plate must define which dimension is its *positive dimension*. The positive dimension is defined as the dimension that corresponds to the dye that has a high fluoresence intensity in all non-empty droplets. The other dimension is defined as the *variable dimension*. For example, assuming the HEX dye is plotted along the X axis and the FAM dye is along the Y axis, a FAM+/FAM+HEX+ plate will have "Y" as its positive dimension because both non-empty clusters have FAM+ droplets. Similarly, a HEX+/FAM+HEX+ plate will have "X" as its positive dimension.

The positive dimension must be set in order to use a `pnpp_experiment`. It is not recommended to use this type directly; instead you should use one of the subtypes (`fam_positive_pnpp` or `hex_positive_pnpp`). If you do use this type directly, you must set the positive dimension with `positive_dim`.

Plates with this type have the following analysis steps: `INITIALIZE`, `REMOVE_FAILURES`, `REMOVE_OUTLIERS`, `REMOVE_EMPTY`, `CLASSIFY`, `RECLASSIFY`.

Plates with this type have the following droplet clusters: `UNDEFINED`, `FAILED`, `OUTLIER`, `EMPTY` (double-negative), `RAIN`, `POSITIVE`, `NEGATIVE`.

See the README for more information on plate types.

See Also

`plate_types`
`fam_positive_pnpp`
`hex_positive_pnpp`
`wildtype_mutant_pnpp`
`positive_dim`

wells_positive
wells_negative
analyze
remove_failures
remove_outliers
remove_empty
classify_droplets
reclassify_droplets

Examples

```
## Not run:
plate <- new_plate(sample_data_dir(), type = plate_types$pnpp_experiment)
type(plate)

## End(Not run)
```

`reset` *Reset a plate*

Description

Reset a ddPCR plate object back to its original state. After resetting a plate, all the analysis progress will be lost, but the original droplet data and plate metadata will be kept. Two common reasons to reset a plate are either to restart the analysis, or to re-analyze the plate as a different plate type.

Usage

```
reset(plate, type, params, keep_type = FALSE, keep_params = FALSE)
```

Arguments

`plate`	A ddPCR plate
`type`	A ddPCR plate type (see `plate_types`)
`params`	List of parameters to set for the plate. Only advanced users should consider using this feature. See `new_plate` for usage.
`keep_type`	If `TRUE` then use keep the same plate type as `plate`
`keep_params`	If `TRUE` then keep the same plate parameters of `plate`

Value

A new unanalyzed ddPCR plate

See Also

plate_types
new_plate

Examples

```
## Not run:
plate <- new_plate(sample_data_dir(), type = plate_types$custom_thresholds)
plate <- reset(plate, type=plate_types$fam_positive_pnpp)

## End(Not run)
```

`sample_data` *Get sample data*

Description

These functions return sample data files or folders and can be used to load ddPCR plates with sample data. They are used primarily in the documentation examples, but you can also use them for learning purposes. There are two sample datasetes: a small dataset and a large dataset. The small dataset contains the full raw data, but the large dataset only includes the processed data because the raw data would be too large.

`sample_data_dir`: get the directory of the small or large sample dataset
`sample_data_file`: get path to one of the data files in the small sample dataset
`sample_results_file`: get path to the results file of the small sample dataset
`sample_plate`: get the ddpcr plate object containing the data of the small or large dataset

Usage

```
sample_data_dir()

sample_data_file()

sample_results_file()

sample_plate(size = c("small", "large"))
```

Arguments

`size`	The dataset to retrieve, either `"small"` or `"large"`

Examples

```
plate1 <- new_plate(dir = sample_data_dir())
plate2 <- new_plate(data_files = sample_data_file(), meta_file = sample_results_file())
plate3 <- sample_plate()
```

save_plate *Save a ddPCR plate*

Description

Saves a plate to a file, including all its data, parameters, and current analysis state. The file can be read back later using load_plate. The file is not human-readable - if you want to save the droplets data or the metadata of a plate, then first retrieve the data using plate_data or plate_meta and save it with write.csv.

Usage

```
save_plate(plate, file)
```

Arguments

plate	Plate object to save.
file	Name of the file where the plate will be saved.

Value

The given plate, unchanged.

See Also

load_plate

Examples

```
plate <- new_plate(sample_data_dir())
save_plate(plate, "myplate")
unlink("myplate.rds")
```

set_default_params *Reset plate parameters to their defaults*

Description

Use this function to reset a ddPCR plate's parameters back to their default values.

Usage

```
set_default_params(plate)
```

Arguments

plate	A ddPCR plate.

Value

The plate with the parameters set to the plate type's default values.

See Also

params

Examples

```
## Not run:
plate <- new_plate(sample_data_dir(), type = plate_types$custom_thresholds)
x_var(plate) <- "VIC"
plate <- set_default_params(plate)

## End(Not run)
```

---

steps *Analysis steps of a ddPCR plate*

Description

Every ddPCR plate type has an ordered set of steps that are run to analyze the data. You can run all the steps with analyze or run the analysis step by step with next_step. The order of the steps in the list is the order in which they are run on the dataset.

Usage

```
steps(plate)
```

Arguments

plate	a ddPCR plate.

Value

A named character vector, where every name is the human-readable name of an analysis step, and every value is the name of the function used to perform the step.

See Also

analyze
next_step

Examples

```
## Not run:
dir <- sample_data_dir()
new_plate(dir) %>% steps
new_plate(dir, plate_types$fam_positive_pnpp) %>% steps

## End(Not run)
```

`subset.ddpcr_plate` *Subsetting a ddPCR plate*

Description

Select specific wells or samples from a ddPCR plate.

Usage

```
## S3 method for class 'ddpcr_plate'
subset(x, wells, samples, targets_ch1, targets_ch2, ...)
```

Arguments

`x`	The ddPCR plate to subset from.
`wells`	Vector or range notation of wells to select (see Range Notation section for more information).
`samples`	Vector of sample names to select.
`targets_ch1`	Vector of target names in channel 1 to select.
`targets_ch2`	Vector of target names in channel 2 to select.
`...`	Ignored

Details

Keeps only data from the selected wells. If sample names are provided instead of well IDs, then any well corresponding to any of the sample names will be kept. Either well IDs or sample names must be provided, but not both.

Value

Plate with data only from the specified wells/samples.

Range notation

The most basic way to select wells is to provide a vector of wells such as `c("B03", "C12")`. When selecting wells, a special range notation is supported to make it easier to select many wells: use a colon (`:`) to specify a range of wells, and use a comma (`,`) to add another well or range. When specifying a range, all wells in the rectangular area between the two wells are selected. For example, `B04:D06` is equivalent to `B04, B05, A05, C04, C05, C06, D04, D05, D06`. You can combine multiple ranges in one selection; see the Examples section below. Note that this notation is only supported for the `wells` parameter, but not for the `samples` parameter.

Examples

```
plate <- new_plate(sample_data_dir())
plate %>% wells_used
plate %>% subset("C01") %>% wells_used
plate %>% subset(c("C01", "F05")) %>% wells_used
plate %>% subset("C01, F05") %>% wells_used
plate %>% subset("C01:F05") %>% wells_used
plate %>% subset("C01:F05, A01") %>% wells_used
plate %>% subset("A01:C03") %>% wells_used
plate %>% subset("A01:C05") %>% wells_used
plate %>% subset("A01, A05:F05") %>% wells_used
plate %>% subset("A01, A05:C05, F05") %>% wells_used
plate %>% subset("A01:A05, C01:C05, F05") %>% wells_used
plate %>% subset(samples = "Dean") %>% wells_used
plate %>% subset(samples = c("Dean", "Mike")) %>% wells_used
```

`thresholds` *Get/set the thresholds*

Description

For ddPCR plates of type `custom_thresholds`, get or set the thresholds that divide the four droplet quadrants.

Usage

```
thresholds(plate)

thresholds(plate) <- value

set_thresholds(plate, value)
```

Arguments

`plate`	A ddPCR plate.
`value`	The new thresholds as a 2-element numeric vector

Value

The current thresholds

See Also

custom_thresholds
x_threshold
y_threshold

Examples

```
## Not run:
plate <- new_plate(sample_data_dir(), type = plate_types$custom_thresholds)
thresholds(plate)
thresholds(plate) <- c(5500, 8000)
set_thresholds(plate, c(5500, 8000))

## End(Not run)
```

---

type *Plate type*

Description

Get the type of a ddPCR plate. See the README for more information on plate types.

Usage

```
type(plate, all = FALSE)
```

Arguments

plate	A ddPCR plate
all	If FALSE, show only the most specific plate type; otherwise, show all inherited (implicit) types as well.

Value

A character vector with the plate type(s).

See Also

plate_types

Examples

```
## Not run:
plate <- new_plate(sample_data_dir(), type = plate_types$fam_positive_pnpp)
type(plate)
type(plate, TRUE)

## End(Not run)
```

`wells_mutant` *Get mutant wells*

Description

After a ddPCR plate of type `wildtype_mutant_pnpp` has been analyzed, get the wells that were deemed as mutant.

Usage

```
wells_mutant(plate)
```

Arguments

`plate`	A ddPCR plate.

Value

Character vector with well IDs of mutant wells

See Also

`wildtype_mutant_pnpp`
`wells_wildtype`

Examples

```
## Not run:
plate <- new_plate(sample_data_dir(), type = plate_types$fam_positive_pnpp) %>% analyze
wells_mutant(plate)

## End(Not run)
```

wells_negative *Get negative wells*

Description

After a ddPCR plate of type pnpp_experiment has been analyzed, get the wells that were not deemed as having mostly positive droplets.

Usage

```
wells_negative(plate)
```

Arguments

plate	A ddPCR plate.

Value

Character vector with well IDs of negative wells

See Also

pnpp_experiment
wells_positive

Examples

```
## Not run:
plate <- new_plate(sample_data_dir(), type = plate_types$pnpp_experiment) %>% analyze
wells_negative(plate)

## End(Not run)
```

wells_positive *Get positive wells*

Description

After a ddPCR plate of type pnpp_experiment has been analyzed, get the wells that were deemed as having mostly positive droplets.

Usage

```
wells_positive(plate)
```

Arguments

plate	A ddPCR plate.

Value

Character vector with well IDs of positive wells

See Also

pnpp_experiment
wells_negative

Examples

```
## Not run:
plate <- new_plate(sample_data_dir(), type = plate_types$pnpp_experiment) %>% analyze
wells_positive(plate)

## End(Not run)
```

---

wells_success *Get successful/failed wells*

Description

Get a list of wells that had successful or failed ddPCR runs. One of the analysis steps for ddPCR plates includes identifying failed wells, which are wells where the ddPCR run was not successful and did not produce useful droplet data.

Usage

```
wells_success(plate)

wells_failed(plate)
```

Arguments

plate	A ddPCR plate

Value

List of wells that had a successful/failed ddPCR run.

See Also

remove_failures

Examples

```
## Not run:
dir <- sample_data_dir()
plate <- new_plate(dir) %>% analyze
plate %>% wells_success
plate %>% wells_failed

## End(Not run)
```

`wells_used` *Get wells used in a ddPCR plate*

Description

Get a list of the wells that have any data in a ddPCR plate.

Usage

```
wells_used(plate)
```

Arguments

`plate`	A ddPCR plate

Value

List of wells that have any data in the given plate.

See Also

`subset.ddpcr_plate`

Examples

```
## Not run:
plate <- new_plate(sample_data_dir(), type = plate_types$custom_thresholds)
wells_used(plate)
plate <- subset(plate, "A01:C05")
wells_used(plate)

## End(Not run)
```

wells_wildtype *Get wildtype wells*

Description

After a ddPCR plate of type `wildtype_mutant_pnpp` has been analyzed, get the wells that were deemed as wildtype.

Usage

```
wells_wildtype(plate)
```

Arguments

`plate`	A ddPCR plate.

Value

Character vector with well IDs of wildtype wells

See Also

`wildtype_mutant_pnpp`
`wells_mutant`

Examples

```
## Not run:
plate <- new_plate(sample_data_dir(), type = plate_types$fam_positive_pnpp) %>% analyze
wells_wildtype(plate)

## End(Not run)
```

well_info *Get metadata info of a well*

Description

Each ddPCR plate has associated metadata that stores infromation for every well. Use this function to retrieve any metadata information for a single well or for a list of wells.

Usage

```
well_info(plate, well_ids, var)
```

Arguments

plate	A ddPCR plate
well_ids	A character vecotr of well IDs denoting the wells to get information for
var	The metadata variable to get (to see a list of all possible metadata variables, use names(plate_meta(plate)))

Value

A character vector with the wanted metadata variable value for each well.

See Also

plate_meta

Examples

```
## Not run:
plate <- new_plate(sample_data_dir(), type = plate_types$custom_thresholds)
well_info(plate, "A01", "drops")

## End(Not run)
```

---

wildtype_mutant_pnpp *Plate type: wildtype/mutant PNPP*

Description

A plate of type wildtype_mutant_pnpp is a subtype of pnpp_experiment that assumes the double-positive cluster denotes wildtype and the other non-empty cluster denotes mutant droplets. There are two plate types that are subtypes of wildtype_mutant_pnpp: fam_positive_pnpp and hex_positive_pnpp. It is not recommended to use this type directly; instead you should use one of the subtypes.

Details

Plates with this type have the following analysis steps: INITIALIZE, REMOVE_FAILURES, REMOVE_OUTLIERS, REMOVE_EMPTY, CLASSIFY, RECLASSIFY.

Plates with this type have the following droplet clusters: UNDEFINED, FAILED, OUTLIER, EMPTY (double-negative), RAIN (not empty but not wildtype nor negative), POSITIVE (wildtype), NEGATIVE (mutant).

See the README for more information on plate types.

See Also

plate_types
fam_positive_pnpp
hex_positive_pnpp
pnpp_experiment
analyze
remove_failures
remove_outliers
remove_empty
classify_droplets
reclassify_droplets

Examples

```
## Not run:
plate <- new_plate(sample_data_dir(), type = plate_types$wildtype_mutant_pnpp)
type(plate)

## End(Not run)
```

---

x_threshold *Get/set the X threshold*

Description

For ddPCR plates of type `custom_thresholds`, get or set the threshold along the X axis that divides the droplet quadrants.

Usage

```
x_threshold(plate)

x_threshold(plate) <- value
```

Arguments

`plate`	A ddPCR plate.
`value`	The new X threshold

Value

The current X threshold

See Also

custom_thresholds
y_threshold
thresholds

Examples

```
## Not run:
plate <- new_plate(sample_data_dir(), type = plate_types$custom_thresholds)
x_threshold(plate)
x_threshold(plate) <- 5500
plot(plate)

## End(Not run)
```

`x_var` *Get/set the X/Y variable (dye name)*

Description

By default, the dye visualized along the X axis is HEX and the dye visualized along the Y axis is FAM. You can use these functions to get or set these values if your plate uses different dyes.

Usage

```
x_var(plate)

y_var(plate)

x_var(plate) <- value

y_var(plate) <- value
```

Arguments

`plate`	A ddPCR plate
`value`	New dye name

Details

The X/Y variables are simply parameters in the plate, which can also be accessed or changed using `params`. You should use these functions to change the X/Y variable rather than changing the parameters directly.

Value

Dye name

See Also

`params`

Examples

```
## Not run:
plate <- new_plate(sample_data_dir())
x_var(plate)
x_var(plate) <- "VIC"
x_var(plate)

## End(Not run)
```

`y_threshold` *Get/set the Y threshold*

Description

For ddPCR plates of type `custom_thresholds`, get or set the threshold along the Y axis that divides the droplet quadrants.

Usage

```
y_threshold(plate)

y_threshold(plate) <- value
```

Arguments

`plate`	A ddPCR plate.
`value`	The new Y threshold

Value

The current Y threshold

See Also

`custom_thresholds`
`x_threshold`
`thresholds`

Examples

```
## Not run:
plate <- new_plate(sample_data_dir(), type = plate_types$custom_thresholds)
y_threshold(plate)
y_threshold(plate) <- 8000
plot(plate)

## End(Not run)
```

Index