# Package 'quicR'

November 15, 2024

**Title** RT-QuIC Data Formatting and Analysis

**Version** 1.0.2

**Description** Designed for the curation and analysis of data generated from real-time quaking-induced conversion (RT-QuIC) assays first described by Atarashi et al. (2011) <doi:10.1038/nm.2294>. 'quicR' calculates useful metrics such as maxpoint ratio: Rowden et al. (2023) <doi:10.1099/vir.0.069906-0>; time-to-threshold: Shi et al. (2013) <doi:10.1186/2051-5960-1-44>; and maximum slope. Integration with the output from plate readers allows for seamless input of raw data into the R environment.

**Imports** dplyr, ggplot2, openxlsx, readxl, reshape2, slider, stats, stringr, tidyr

**License** GPL-3

**Encoding** UTF-8

**RoxygenNote** 7.3.2

**Suggests** testthat (>= 3.1.0)

**Config/testthat/edition** 3

**NeedsCompilation** no

**Author** Gage Rowden [aut, cre] (<https://orcid.org/0000-0002-7517-0480>)

**Maintainer** Gage Rowden <gage.rowden1145@gmail.com>

**Repository** CRAN

**Date/Publication** 2024-11-15 09:00:05 UTC

## Contents

---

`add_reps` *Add replicates*

---

## Description

Adds replicate information to the sample IDs. Well IDs should be formatted like so: A4, B9, H11, J24

## Usage

```
add_reps(df, sep = "_")
```

## Arguments

| | |
|---|---|
| `df` | A dataframe containing two columns for well IDs and Sample IDs |
| `sep` | a character string to separate the terms. |

## Value

A dataframe with replicate numbers pasted to the Sample IDs

---

`BMG_format` *Format Table for BMG Sample ID Import*

---

## Description

BMG_format accepts a plate layout .CSV file and formats the Sample IDs into a format which can be easily imported into the BMG control software.

## Usage

```
BMG_format(
  file,
  save_path = "./",
  save_name = "formatted.txt",
  write_file = FALSE
)
```

### Arguments

| | |
|---|---|
| file | A .CSV file containing the plate layout of Sample IDs. |
| save_path | The path to the directory that you want the file saved. |
| save_name | The name of the output file. Should have the ".txt" extension. |
| write_file | Logical. If true, function will write a .txt file; otherwise it will return a character vector. |

### Value

A text file containing information for import into the BMG control software.

### Examples

```
layout_file <- system.file(
  "extdata/BMG_formatting",
  file = "plate_layout.csv",
  package = "quicR"
  )
BMG_format(layout_file)
```

---

## calculate_MPR *Calculate the Maxpoint Ratio*

---

### Description

Maxpoint ratio is defined as the maximum relative fluorescence divided by the background fluorescence.

### Usage

```
calculate_MPR(data, start_col = 3, data_is_norm = FALSE)
```

### Arguments

| | |
|---|---|
| data | A dataframe containing the real-time fluorescence data. |
| start_col | Integer, the column at which the background fluorescence should be read. |
| data_is_norm | Logical, if the data has not been normalized, will make a call to normalize_RFU. |

### Value

A vector containing MPR values.

## Examples

```
# This test takes >5 sec

  file <- system.file(
    "extdata/input_files",
    file = "test.xlsx",
    package = "quicR"
    )
  df_ <- quicR::get_real(file)[[1]]
  print(calculate_MPR(df_))
```

---

`calculate_MS` *Calculate Maximum Slope*

---

## Description

Uses a sliding window to calculate the slope of real-time reads.

## Usage

```
calculate_MS(data, window = 3)
```

## Arguments

| | |
|---|---|
| `data` | A dataframe containing real-time reads. It is recommended to use a dataframe made from normalize_RFU. |
| `window` | Integer designating how wide you want the sliding window to be for calculating the moving average slope. |

## Value

A dataframe containing the real-time slope values.

## Examples

```
# This test takes >5 sec

  file <- system.file(
    "extdata/input_files",
    file = "rt_data.csv",
    package = "quicR"
)
df_ <- read.csv(file, check.names = FALSE)
calculate_MS(df_)
```

`calculate_TtT` *Calculate Time to Threshold*

## Description

Calculates the time required to reach a defined threshold.

## Usage

```
calculate_TtT(data, threshold, start_col = 3, run_time = 48)
```

## Arguments

| | |
|---|---|
| `data` | A dataframe containing real-time RT-QuIC data. |
| `threshold` | A numeric value defining the threshold. |
| `start_col` | The column containing the starting position of the real-time data. |
| `run_time` | The time in hours that the assay ran. |

## Value

A vector containing the times to threshold

## Examples

```
# This test takes >5 sec

file <- system.file(
"extdata/input_files",
file = "test.xlsx",
package = "quicR"
)
df_ <- get_real(file)[[1]] |>
  normalize_RFU()
calculate_TtT(df_, threshold = 2)
```

## convert_tables *Convert tables into a single column in a dataframe.*

### Description

Accepts a table or matrix or a list of tables and matrices and converts them into dataframe columns.

### Usage

```
convert_tables(tab)
```

### Arguments

| | |
|---|---|
| tab | A table/matrix or a list of tables/matrices. |

### Value

A dataframe column.

### Examples

```
file <- system.file(
"extdata/input_files",
file = "test.xlsx",
package = "quicR"
)
tabs <- organize_tables(file)
convert_tables(tabs)
```

## get_meta *Retrieve the BMG metadata*

### Description

Takes the Excel file exported from MARS and compiles the metadata in the header.

### Usage

```
get_meta(file)
```

### Arguments

| | |
|---|---|
| file | The Excel file exported from MARS. |

## Value

A dataframe containing the Meta_ID and Meta_info

## Examples

```
file <- system.file(
"extdata/input_files",
file = "test.xlsx",
package = "quicR"
)
get_meta(file)
```

---

`get_real` *Get Real-Time RT-QuIC Fluorescence Data*

---

## Description

Accepts an Excel file or a dataframe of real-time RT-QuIC data.

## Usage

```
get_real(file, ordered = FALSE)
```

## Arguments

| | |
|---|---|
| `file` | Either an Excel file or a dataframe. |
| `ordered` | Logical, if true, will organize the columns by sample ID. |

## Value

A list of dataframes containing the formatted real-time data.

## Examples

```
file <- system.file(
"extdata/input_files",
file = "test.xlsx",
package = "quicR"
)
get_real(file)
```

## get_wells *Get the Wells Used in the RT-QuIC Run.*

### Description

Returns the well IDs used in the plate.

### Usage

```
get_wells(file)
```

### Arguments

| | |
|---|---|
| file | Excel file exported from MARS |

### Value

A vector containing well IDs.

### Examples

```
file <- system.file(
"extdata/input_files",
file = "test.xlsx",
package = "quicR"
)
get_wells(file)
```

## normalize_RFU *Normalize Fluorescence*

### Description

Normalizes the real-time RT-QuIC data against the background fluorescence of a defined cycle. All cycles are divided by the fluorescent value of the defined cycle.

### Usage

```
normalize_RFU(df, bg_cycle = 4)
```

### Arguments

| | |
|---|---|
| df | A dataframe made from get_real. |
| bg_cycle | The cycle used for background fluorescence |

## Value

A dataframe containing real-time normalized fluorescence values.

## Examples

```
# This test takes >5 sec

file <- system.file(
  "extdata/input_files",
  file = "test.xlsx",
  package = "quicR"
)
df_ <- get_real(file)[[1]]

# Export the tables in the first sheet of the file.
dic <- quicR::organize_tables(file)

# Apply the column names.
colnames(df_) <- cbind("Time", convert_tables(dic)$`Sample IDs` |> t())

# Normalize the raw data against the background reading.
normalize_RFU(df_)
```

---

`organize_tables` *Organize MARS Tables*

---

## Description

Extracts the tables from the microplate view sheet in the MARS Excel file and adds each table to a list.

## Usage

```
organize_tables(file, plate = 96)
```

## Arguments

| | |
|---|---|
| `file` | An Excel file exported from MARS. |
| `plate` | Integer either 96 or 384 to denote microplate type. |

## Value

A list containing tibbles.

## Examples

```
file <- system.file(
"extdata/input_files",
file = "test.xlsx",
package = "quicR"
)
organize_tables(file)
```

---

plate_view *Real-Time Plate View*

---

## Description

Converts the real-time data into a ggplot figure. The layout is either 8x12 or 16x24 for 96- and 384-well plates, respectively.

## Usage

```
plate_view(df, meta, plate = 96)
```

## Arguments

| | |
|---|---|
| df | Real-time dataframe |
| meta | Dataframe containing well IDs and Sample IDs to title each facet. |
| plate | Integer either 96 or 384 to denote microplate type. |

## Value

A ggplot object

## Examples

```
# This test takes >5 sec

file <- system.file(
"extdata/input_files",
file = "test.xlsx",
package = "quicR"
)

tab <- organize_tables(file)
IDs <- quicR::convert_tables(tab)[["Sample IDs"]] |>
na.omit()

# Get the real-time data.
df_ <- get_real(file, ordered = FALSE)[[1]]
```

```
# Set the time column as the df index.
rownames(df_) <- df_[, 1]

# Remove the time column and ID row.
df_ <- df_[, -1]

# Get the wells used in the run.
wells <- get_wells(file)

# Take the metadata and apply it into a dataframe for the plate_view function.
sample_locations <- cbind(wells, IDs) |>
  stats::na.omit()

# Wrap the text if it is too long.
sample_locations <- sample_locations |>
  dplyr::mutate(IDs = ifelse(stringr::str_length(IDs) > 12, gsub(" ", "\n", IDs), IDs))

# Make the plate view figure.
plate_view(df_, sample_locations, plate = 96)
```

---

`separate_raw` *Separate Real-Time Data into separate dataframes.*

---

## Description

If multiple real-time reads were exported from MARS, separate_raw will parse them out and separate them. It will also export to an Excel file with each real-time data having its own sheet.

## Usage

```
separate_raw(file, num_rows, export_name)
```

## Arguments

| | |
|---|---|
| `file` | An Excel file exported from MARS. |
| `num_rows` | Number of rows in the header to ignore. |
| `export_name` | The name of the original file or an orignal name. |

## Value

An Excel file with separated raw real-time data.

# Index