quicR: An R Package for Real-Time Quaking Induced Conversion (RT-QuIC) Assays

Gage Rowden Peter Larsen

Abstract

Real-time quaking induced conversion (RT-QuIC) has quickly become an emerging diagnostic tool for protein misfolding disorders such as Creutzfeldt-Jakob disease and Parkinson's disease. Given that the technology is still relatively new, academic and industry standards have yet to be established. 'quicR' was developed to fill this lack of standardization by providing functions for data curation, analysis, and vizualiztion.

Introduction

Real-time quaking induced conversion is a diagnostic assay that converts recombinant protein substrate into a misfolded aggregate in the presence of a misfolded seed (Atarashi et al. 2011). Given the recent development and novelty of the assay, academic and industry standards for analysis have yet to be established. 'quicR' is an attempt to provide an open-source library dedicated to RT-QuIC assays.

While standard metrics for determining a diagnosis have not been universally established, there are certain metrics that many research groups have found useful. These include time-to-threshold (TtT)(Orrú et al. 2015), rate of amyloid formation (RAF)(Gallups and Harms 2022), maxpoint ratio (MPR)(Rowden et al. 2023), and maximum slope (MS)(Henderson et al. 2015). All together, these metrics provide insight into the general kinetics of an RT-QuIC reaction, and can be used together to draw a more robust diagnostic decision.

quicR also provides visualization options using ggplot2 (Wickham 2016) on the backend. Figures generated by quicR functions can be manipulated using the ggplot2 "+" syntax.

Methods

This package requires the following dependencies: dplyr, ggplot2, janitor, openxlsx, readxl, reshape2, slider, stats, stringr, tidyr. Because the MARS software (BMG Labtech, Ortenberg, Germany) exports data as an Excel workbook, the packages, openxlsx and readxl, were fundamental to performing downstream handling. The tidyverse packages (dplyr, ggplot2, stringr,

and tidyr), were vital for writing easy-to-read code and for data visualization. The janitor package has useful functions for performing quick data cleaning

Development

The quicR package was developed to address the need for efficient data conversion and analysis of RT-QuIC data. The functions were designed with usability and reproducibility in mind, ensuring compatibility between multiple labs. Currently, the package accepts data exported from the MARS software as an Excel workbook.

The functionality in this package revolves around data curation, metric calculations, and visualization.

Implementation

Input of Sample IDs into MARS

MARS allows input of a TXT file containing sample IDs, dilution factors, and their well locations. This file is uniquely formatted, and not easily reproduced manually. The function, "BMG_format", allows for input of a CSV file containing the plate layout, and exports the formatted TXT file.

Example CSV File Plate Layout

	1	2	3	4	5	6	7	8	9	10	11	12
A	P	S01	S02	S03	S04	S05	S06	S07	S08	S09	S10	S11
B	P	S01	S02	S03	S04	S05	S06	S07	S08	S09	S10	S11
C	P	S01	S02	S03	S04	S05	S06	S07	S08	S09	S10	S11
D	P	S01	S02	S03	S04	S05	S06	S07	S08	S09	S10	S11
E	N	S01	S02	S03	S04	S05	S06	S07	S08	S09	S10	S11
F	N	S01	S02	S03	S04	S05	S06	S07	S08	S09	S10	S11
G	N	S01	S02	S03	S04	S05	S06	S07	S08	S09	S10	S11
H	N	S01	S02	S03	S04	S05	S06	S07	S08	S09	S10	S11

Formatted Plate Layout for MARS Input

The function, "BMG_format", includes the logical argument "write_file". If TRUE, it will create a TXT file. The path can be given to the "save_path" argument, and the file name can be supplied to the "save_name" argument.

```
BMG_format(sample_file, write_file = TRUE)
```

```
A1  P      P
B1  P      P
C1  P      P
D1  P      P
E1  N      N
F1  N      N
G1  N      N
H1  N      N
A2  X1     S01
B2  X1     S01
C2  X1     S01
D2  X1     S01
E2  X1     S01
F2  X1     S01
G2  X1     S01
H2  X1     S01
```

Data Cleaning and Transformation

The MARS software (BMG Labtech, Ortenberg, Germany) exports real-time data as an Excel workbook. Typically, the first sheet in the workbook will include microplate views of both raw data and metadata. The two tables that are most relevant on this sheet are the "Sample IDs" and the "Dilutions" tables (if included). For much of the downstream analysis, it is crucial the the "Sample IDs" table was exported. If there is no table, the user can simply add it manually.

Retrieving Metadata

The metadata is defined as either sample-dependent or -independent. Sample-dependent metadata includes information such as sample IDs and dilution factors, whereas sample-independent metadata includes the date, time, reaction ID, etc.

The dependent metadata can be retrieved using the "organize_tables" and "convert_tables" functions. The former returns a list of tables, and the ladder converts each table into a column in a single data frame.

```
tabs <- organize_tables(file)[-2]
```

Sample IDs

	1	2	3	4	5	6	7	8	9	10	11	12
A	P	S01	S02	S03	S04	S05	S06	S07	S08	S09	S10	S11
B	P	S01	S02	S03	S04	S05	S06	S07	S08	S09	S10	S11
C	P	S01	S02	S03	S04	S05	S06	S07	S08	S09	S10	S11
D	P	S01	S02	S03	S04	S05	S06	S07	S08	S09	S10	S11
E	N	S01	S02	S03	S04	S05	S06	S07	S08	S09	S10	S11
F	N	S01	S02	S03	S04	S05	S06	S07	S08	S09	S10	S11
G	N	S01	S02	S03	S04	S05	S06	S07	S08	S09	S10	S11
H	N	S01	S02	S03	S04	S05	S06	S07	S08	S09	S10	S11

Dilutions

	1	2	3	4	5	6	7	8	9	10	11	12
A	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000
B	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000
C	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000
D	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000
E	1000	10000	10000	10000	10000	10000	10000	10000	10000	10000	10000	10000
F	1000	10000	10000	10000	10000	10000	10000	10000	10000	10000	10000	10000
G	1000	10000	10000	10000	10000	10000	10000	10000	10000	10000	10000	10000
H	1000	10000	10000	10000	10000	10000	10000	10000	10000	10000	10000	10000

```
tabs <- convert_tables(tabs)
tabs %>% head(12) %>% kable(row.names = FALSE)
```

Sample IDs	Dilutions
P	1000
S01	1000
S02	1000
S03	1000
S04	1000
S05	1000
S06	1000
S07	1000
S08	1000
S09	1000
S10	1000
S11	1000

Sample Locations

Samples locations can be extracted based on their well ID. The “get_sample_locations” function accepts additional arguments if dilution factors were exported from MARS. The “dilution_fun” argument will supply a function for transforming the dilution factors (e.g. if the user would want to perform a log transformation).

```
get_sample_locations(
  file,
  dilution_bool = TRUE,
  dilution_fun = function(x)
    -log10(x),
  sep = " "
)
```

V1	IDs
A01	P -3
A02	S01 -3
A03	S02 -3
A04	S03 -3
A05	S04 -3
A06	S05 -3
A07	S06 -3
A08	S07 -3
A09	S08 -3
A10	S09 -3
A11	S10 -3
A12	S11 -3

Sample-Independent Metadata

The independent metadata can be retrieved using the "get_meta" function. This data is included in the header of the excel workbook.

```
get_meta(file)
```

Meta_ID	Meta_info
User	USER
Path	C:/Program Files (x86)/BMG/Omega/User/Data
Test ID	54
Test Name	RT-QuIC Plate Mode
Date	4/24/2024
Time	2:41:47 PM
ID1	20240424_r4_PRC_Oral_Swabs
Fluorescence (FI)	NA

Retrieving and Manipulating Raw Data

The raw data is typically found on the second sheet of the Excel workbook. The raw real-time data can be retrieved using the "get_real" function. The logical argument, "ordered", indicates whether the user would prefer the columns to be ordered by well or by sample ID. By default, it is FALSE which will order the data by well. This should almost always be the case for easier integration with other downstream functions. Additionally, since there can be more than one instance of real-time data (depending on if the user added some calculations in MARS), "get_real" returns a list of dataframes. Therefore, the output should be indexed to access the data frame of interest.

Retrieve Raw Data

```
get_real(file)[[1]]
```

Time	sample_x1	sample_x2	sample_x3	sample_x4	sample_x5	sample_x6	sample_x7
0	813	664	670	643	685	396	677
0.75	539	477	470	454	469	372	480
1.5	564	476	484	470	486	375	481
2.25	600	480	481	473	488	378	494
3	622	476	478	474	487	368	486
3.75	618	476	488	470	502	376	494
4.5	623	482	486	479	484	368	499
5.25	629	476	494	478	497	379	501
6	618	483	490	477	493	365	506
6.75	626	485	495	490	502	368	507
7.5	624	482	495	479	502	368	508

Transpose Raw Data

This data is structured such that each sample is its own column (variable) and each row (observation) is a time point. While this format is technically correct, a transposed format is more ideal for some downstream manipulation. This operation is performed using the function, "transpose_real". After transposition, each time point is an individual column (variable), and each sample is an individual row (observation).

```
get_real(file)[[1]] %>% transpose_real()
```

Sample IDs	0	0.75	1.5	2.25	3	3.75	4.5	5.25	6	6.75	7.5
sample_x1	813	539	564	600	622	618	623	629	618	626	624
sample_x2	664	477	476	480	476	476	482	476	483	485	482
sample_x3	670	470	484	481	478	488	486	494	490	495	495
sample_x4	643	454	470	473	474	470	479	478	477	490	479
sample_x5	685	469	486	488	487	502	484	497	493	502	502
sample_x6	396	372	375	378	368	376	368	379	365	368	368
sample_x7	677	480	481	494	486	494	499	501	506	507	508

Normalize Raw Data

The function "normalize_RFU" will convert the raw data into a background normalized data set. The function includes two additional arguments, "bg_cycle" (the cycle which will be used as the background fluorescence value) and "transposed" (if FALSE, will make a call to the "transpose_real" function). Note that the fourth time point is all "1's" since this was designated the background cycle.

```
get_real(file)[[1]] %>% normalize_RFU(transposed = FALSE)
```

Sample IDs	0	0.75	1.5	2.25	3	3.75	4.5	5.25	6	6.75	7.5
sample_x1	1.35	0.90	0.94	1	1.04	1.03	1.04	1.05	1.03	1.04	1.04
sample_x2	1.38	0.99	0.99	1	0.99	0.99	1.00	0.99	1.01	1.01	1.00
sample_x3	1.39	0.98	1.01	1	0.99	1.01	1.01	1.03	1.02	1.03	1.03
sample_x4	1.36	0.96	0.99	1	1.00	0.99	1.01	1.01	1.01	1.04	1.01
sample_x5	1.40	0.96	1.00	1	1.00	1.03	0.99	1.02	1.01	1.03	1.03
sample_x6	1.05	0.98	0.99	1	0.97	0.99	0.97	1.00	0.97	0.97	0.97
sample_x7	1.37	0.97	0.97	1	0.98	1.00	1.01	1.01	1.02	1.03	1.03

Calculations

There are three analytical metrics with dedicated functions: time-to-threshold (TtT), max-point ratio (MPR), and maximum slope (MS). The rate of amyloid formation does not have a designated function since it is simply the inverse of the time-to-threshold. Each function below accepts input from the "transpose_real" or the "normalize_RFU" functions.

Time-to-threshold

TtT is calculated using the "calculate_TtT" function. The function must be supplied a threshold; default value is 2 (i.e. twice the background fluorescence if the data is normalized). A starting column should also be given as an integer; default value is 3. This is essentially asking how many columns of metadata are included before the fluorescence reads begin.

TtT is calculated by iterating through each row and checking if a value is greater than the threshold. If the value is greater, the slope of the previous time-point to the current time-point is calculated, and the time intersection of the current read is returned.

Maxpoint Ratio

MPR is calculated by the "calculate_MPR" function. Data must be normalized in order to derive this metric. In a normalized data set, the MPR is simply the maximum value achieved during the run. Raw data can be passed to this function, but the argument, "data_is_norm",

must be set to TRUE. This will pass the raw data to “normalize_RFU” before calculating the MPR values.

Maximum Slope

MS is calculated by the “calculate_MS” function. The function iterates through each row using a rolling window which can be adjusted (default value is 3). Given the window size, the slope is calcualated based on the range of the window. The MS is simply the largest slope value recorded.

```
samples <- tabs$`Sample IDs`
dilutions <- tabs$Dilutions

df_norm <- get_real(file) %>% normalize_RFU()

data.frame("Sample IDs" = samples, check.names = FALSE) %>%
  mutate(
    Dilutions = -log10(dilutions),
    # Maxpoint Ratio
    MPR = calculate_MPR(df_norm, start_col = 3, data_is_norm = TRUE),
    # Max Slope
    MS = calculate_MS(df_norm, data_is_norm = TRUE),
    # Time to Threshold
    TtT = calculate_TtT(df_norm, threshold = 2, start_col = 3),
    # Rate of Amyloid Formation
    RAF = 1 / TtT
  )
```

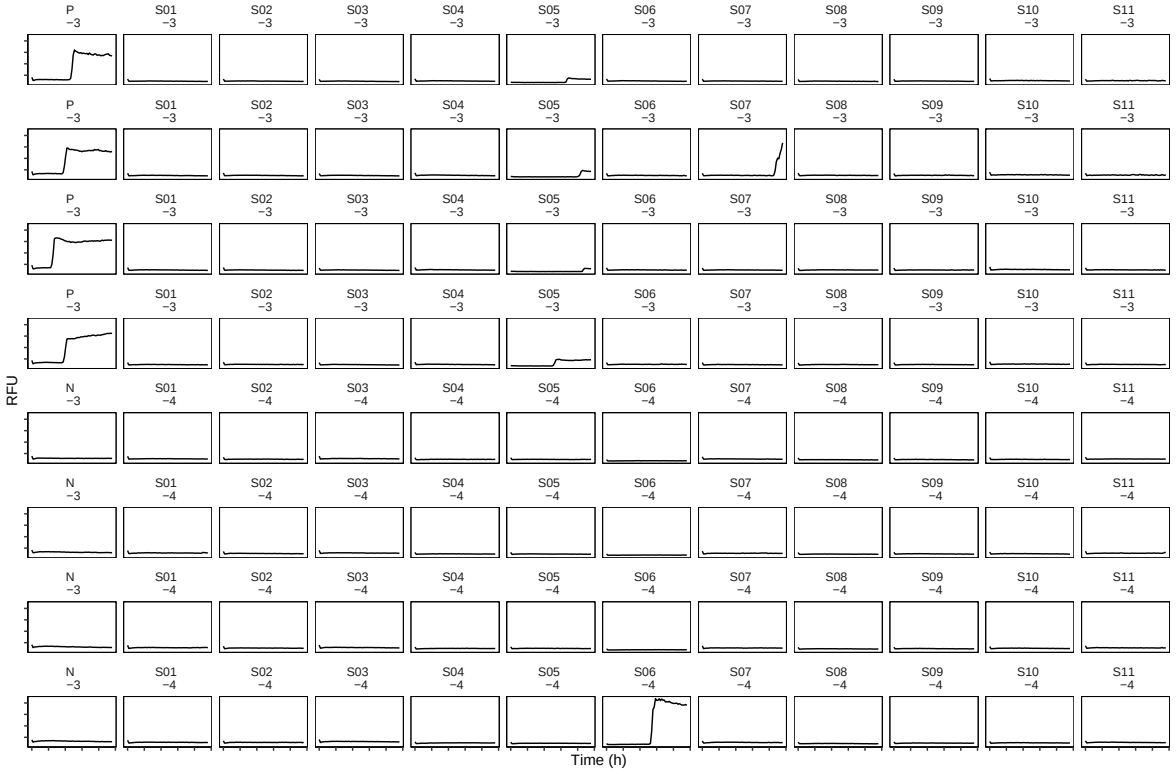
Sample IDs	Dilutions	MPR	MS	TtT	RAF
P	-3	5.351667	1.8362963	23.71003	0.0421763
S01	-3	1.045833	0.1740739	48.00000	0.0208333
S02	-3	1.056133	0.1848000	48.00000	0.0208333
S03	-3	1.069767	0.1775898	48.00000	0.0208333
S04	-3	1.075820	0.1967215	48.00000	0.0208333
S05	-3	1.992063	0.4503233	48.00000	0.0208333
S06	-3	1.066802	0.1772378	48.00000	0.0208333
S07	-3	1.068893	0.1753653	48.00000	0.0208333
S08	-3	1.096491	0.1793372	48.00000	0.0208333
S09	-3	1.055785	0.1689624	48.00000	0.0208333
S10	-3	1.082852	0.1815457	48.00000	0.0208333
S11	-3	1.116700	0.1547060	48.00000	0.0208333

Visualization

Plate View

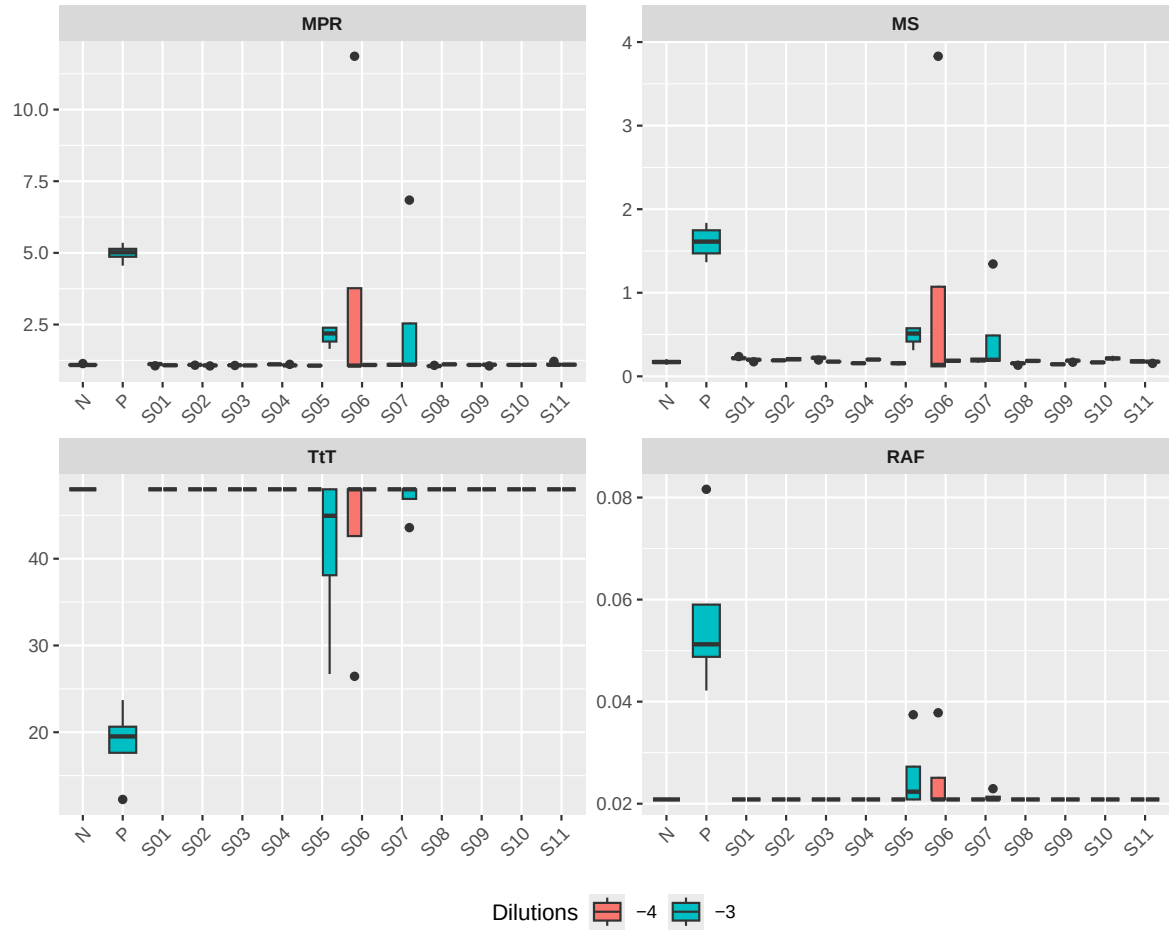
The “plate_view” function requires *un-transposed* data and sample locations as arguments. It also has an argument for plate type which will either be 96 or 384.

```
sample_locations <- get_sample_locations(
  file,
  dilution_bool = TRUE,
  dilution_fun = function(x) -log10(x)
)

plate_view(df_, sample_locations)
```



Summary Plots

```
df_analyzed %>%
  melt(id.vars = c("Sample IDs", "Dilutions")) %>%
  mutate_at("Dilutions", as.factor) %>%

  ggplot(aes(`Sample IDs`, value, fill = Dilutions)) +
    geom_boxplot() +
    facet_wrap(~variable, scales = "free") +
    theme(
      legend.position = "bottom",
      strip.text = element_text(face = "bold"),
      axis.text.x = element_text(angle = 45, hjust = 1, vjust = 1),
      axis.title = element_blank()
    )
```



Usage

Installation

```
# Latest CRAN release
install.packages("quicR")

# Development version
install.packages("devtools")
library(devtools)

install_github("gage1145/quicR")
```

Validation & Performance

Discussion

Conclusion

quicR provides improved and standardized methods for analyzing RT-QuIC data.

Acknowledgments

References

Atarashi, Ryuichiro, Kazunori Sano, Katsuya Satoh, and Noriyuki Nishida. 2011. "Real-Time Quaking-Induced Conversion: A Highly Sensitive Assay for Prion Detection." *Prion*. Taylor & Francis. https://doi.org/10.4161/pri.5.3.16893.

Gallups, Nicole J., and Ashley S. Harms. 2022. "'Seeding' the Idea of Early Diagnostics in Synucleinopathies." *Brain* 145 (April): 418–19. https://doi.org/10.1093/BRAIN/AWAC062.

Henderson, Davin M., Kristen A. Davenport, Nicholas J. Haley, Nathaniel D. Denkers, Candace K. Mathiason, and Edward A. Hoover. 2015. "Quantitative Assessment of Prion Infectivity in Tissues and Body Fluids by Real-Time Quaking-Induced Conversion." *Journal of General Virology* 96 (January): 210–19. https://doi.org/10.1099/vir.0.069906-0.

Orrú, Christina D., Bradley R. Groveman, Andrew G. Hughson, Gianluigi Zanusso, Michael B. Coulthart, and Byron Caughey. 2015. "Rapid and Sensitive RT-QuIC Detection of Human Creutzfeldt-Jakob Disease Using Cerebrospinal Fluid." *mBio* 6 (January). https://doi.org/10.1128/mBio.02451-14.

Rowden, Gage R., Catalina Picasso-Risso, Manci Li, Marc D. Schwabenlander, Tiffany M. Wolf, and Peter A. Larsen. 2023. "Standardization of Data Analysis for RT-QuIC-Based Detection of Chronic Wasting Disease." *Pathogens* 12 (February): 309. https://doi.org/10.3390/PATHOGENS12020309/S1.

Wickham, Hadley. 2016. *Ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York. https://ggplot2.tidyverse.org.