Package 'SeroTrackR'

May 22, 2026

Type Package

Title Serology-Based Data Analysis and Visualization

Version 1.1.0

Description Data wrangling and cleaning, quality control checks and implementation of machine learning classification algorithm.

License CC BY 4.0

Encoding UTF-8

Depends R (>= 4.1.0)

Imports dplyr, drc, forcats, ggplot2, here, janitor, kableExtra, knitr, magrittr, openxlsx, parsnip, purrr, ranger, readxl, rmarkdown, stats, stringr, tidyr, tidyselect, utils, workflows

URL https://github.com/dionnecargy/SeroTrackR, https://dionnecargy.github.io/SeroTrackR/

BugReports https://github.com/dionnecargy/SeroTrackR/issues

Suggests glue, htmltools, httr, jsonlite, readr, rsconnect, shiny.fluent, tidyverse, zoo

VignetteBuilder knitr

Config/roxygen2/version 8.0.0

NeedsCompilation no

Author Dionne Argyropoulos [aut, cre]

Maintainer Dionne Argyropoulos <argyropoulos.d@wehi.edu.au>

Repository CRAN

Date/Publication 2026-05-22 11:40:14 UTC

Contents

`.check_platform` *Check Platform*

Description

This function checks the platform the user has input and whether it aligns with the correct format as expected. Will report error if NOT aligned.

Usage

```
.check_platform(raw_data, platform, file_name)
```

Arguments

`raw_data`	String with the raw data path.
`platform`	"magpix", "bioplex" or "intelliflex".
`file_name`	String with the raw data filename (for error messaging).

Value

TRUE: if platform == file format, ERROR message when platform does not equal file format.

Author(s)

Dionne Argyropoulos

Examples

```
your_raw_data <- system.file("extdata", "example_MAGPIX_plate1.csv", package = "SeroTrackR")
.check_platform(raw_data = your_raw_data, platform = "magpix", file_name = basename(your_raw_data))
```

.clean_bioplex *Helper function to process bioplex data*

Description

Helper function to process bioplex data

Usage

```
.clean_bioplex(df)
```

Arguments

df	Output from '.read_luminex.file()'

Value

Cleaned data fame

Author(s)

Dionne Argyropoulos

Examples

```
your_raw_data <- system.file("extdata", "example_BioPlex_plate1.xlsx", package = "SeroTrackR")
df            <- .read_luminex_file(your_raw_data)
results       <- .clean_bioplex(df)
```

.clean_luminex *Helper function to process luminex (Magpix/Intelliflex) data*

Description

Helper function to process luminex (Magpix/Intelliflex) data

Usage

```
.clean_luminex(df, row1, row2)
```

Arguments

df	Raw luminex file
row1	Leading row to subset
row2	Final row to subset

Value

Cleaned data fame

Author(s)

Dionne Argyropoulos

Examples

```
your_raw_data <- system.file("extdata", "example_MAGPIX_plate1.csv", package = "SeroTrackR")
df            <- .read_luminex_file(your_raw_data)
cfg           <- .magpix_version_config("4.2")

row1         <- which(df$xPONENT == "Median")
row2         <- which(df$xPONENT == "Net MFI")

results      <- .clean_luminex(df, row1, row2)
```

.convert_dilution_to_mfi *Convert known dilution to mfi from fitted standard curve*

Description

Convert dilution to predicted mfi using known standard curve fit.

Usage

```
.convert_dilution_to_mfi(dilution, params)
```

Arguments

dilution	Known dilution of samples
params	Known parameters for five parameter logistic fit.

Value

Returns the predicted mfi of a sample with known dilution.

Author(s)

Eamon Conway

Examples

```
# This function is typically called internally by higher-level workflows.
# Below is a minimal runnable example using dummy parameters.

# Five-parameter logistic model typically expects parameters in the order:
# a, b, c, d, e  (e often log-transformed)
dummy_params <- c(a = 10000, b = 1.2, c = 0.05, d = 50, e = log(0.01))

# Example dilution value
dilution_example <- 0.1

# Predict MFI from the dummy standard curve
.convert_dilution_to_mfi(dilution_example, dummy_params)
```

.convert_mfi_to_dilution

Convert mfi to dilution using known standard curve fit.

Description

Convert mfi to dilution using known standard curve fit.

Usage

```
.convert_mfi_to_dilution(mfi, params, min_relative_dilution)
```

Arguments

`mfi`	Known mfi of samples
`params`	Known parameters for five parameter logistic fit.
`min_relative_dilution`	Known minimum value of dilution in the standard curve. Relative means setting S1 to a dilution/RAU/concentration of 1.

Value

Returns the dilution of each sample in mfi.

Author(s)

Eamon Conway

Examples

```
# This function is typically used within larger analysis pipelines.
# Below is a minimal runnable example using dummy values.

# Dummy five-parameter logistic fit parameters:
# a, b, c, d, e  (with e on the log scale)
# Additional placeholders (f, g) included so params[6] and params[7] exist.
dummy_params <- c(a = 10000, b = 1.2, c = 0.05, d = 50, e = log(0.01),
                  f = -5, g = 5)

# Example MFI value
mfi_example <- 1500

# Minimum relative dilution allowed
min_rel_dil <- 1

# Convert MFI to dilution
.convert_mfi_to_dilution(mfi_example, dummy_params, min_rel_dil)
```

.convert_mfi_to_dilution_no_bounds

Convert mfi to dilution using known standard curve fit and no bounds

---

Description

Convert mfi to dilution using known standard curve fit and no bounds unless you are below the asymptote of the standard curve. In this situation we set your value to min_relative_dilution. I dunno argue?

Usage

```
.convert_mfi_to_dilution_no_bounds(mfi, params, min_relative_dilution)
```

Arguments

`mfi`	Known mfi of samples
`params`	Known parameters for five parameter logistic fit.
`min_relative_dilution`	Known minimum value of dilution in the standard curve. Relative means setting S1 to a dilution/RAU/concentration of 1.

Value

Returns the dilution of each sample in mfi.

Author(s)

Eamon Conway

Examples

```
# This function is generally called inside higher-level analysis workflows.
# Below is a minimal self-contained example using dummy values.

# Dummy five-parameter logistic fit parameters:
# a, b, c, d, e  (with e typically supplied on the log scale)
dummy_params <- c(a = 10000, b = 1.2, c = 0.05, d = 50, e = log(0.01))

# Example MFI value
mfi_example <- 1500

# Minimum relative dilution from the standard curve
min_rel_dil <- 1

# Convert MFI to dilution without bounds
.convert_mfi_to_dilution_no_bounds(mfi_example, dummy_params, min_rel_dil)
```

.convert_mfi_to_dilution_no_lower_bound

Convert mfi to dilution using known standard curve fit and no lower bound

---

Description

Convert mfi to dilution using known standard curve fit and no lower bound unless you are below the asymptote of the standard curve. In this situation we set your value to min_relative_dilution. I dunno argue?

Usage

```
.convert_mfi_to_dilution_no_lower_bound(mfi, params, min_relative_dilution)
```

Arguments

`mfi`	Known mfi of samples
`params`	Known parameters for five parameter logistic fit.
`min_relative_dilution`	Known minimum value of dilution in the standard curve. Relative means setting S1 to a dilution/RAU/concentration of 1.

Value

Returns the dilution of each sample in mfi.

Author(s)

Eamon Conway

Examples

```
# This function is usually called inside higher-level analysis steps.
# Below is a minimal runnable example using dummy values.

# Dummy five-parameter logistic fit parameters:
# a, b, c, d, e  (with e typically on the log scale)
dummy_params <- c(a = 10000, b = 1.2, c = 0.05, d = 50, e = log(0.01), f = 0, g = 5)

# Example MFI value
mfi_example <- 1500

# Minimum relative dilution from the standard curve
min_rel_dil <- 1

# Convert MFI to dilution without applying a lower bound
.convert_mfi_to_dilution_no_lower_bound(mfi_example, dummy_params, min_rel_dil)
```

`.extract_luminex_sections` *Helper function to process luminex sections*

Description

Helper function to process luminex sections

Usage

```
.extract_luminex_sections(df, cfg, plt)
```

Arguments

`df`	String with the raw data path.
`cfg`	Magpix version output of .magpix_version_config().
`plt`	Platform (magpix, intelliflex)

Value

List of data_raw, results, counts, blanks, stds, run

Author(s)

Dionne Argyropoulos

Examples

```
your_raw_data <- system.file("extdata", "example_MAGPIX_plate1.csv", package = "SeroTrackR")
df            <- .read_luminex_file(your_raw_data)
cfg           <- .magpix_version_config("4.2")
section       <- .extract_luminex_sections(df, cfg, "magpix")
```

`.fit_standard_curve` *Fit a standard curve to known mfi and dilution values.*

Description

We wish to convert the standard curve samples to a five parameter logistic curve. This function takes those values and calls optim to determine the fit.

Usage

```
.fit_standard_curve(mfi, dilution, control = NULL)
```

Arguments

`mfi`	Known mfi of samples
`dilution`	Known dilution of samples
`control`	Optional list of control parameters for the underlying call to optim.

Value

standard curve log logistic

Author(s)

Eamon Conway

Examples

```
# This function is typically called within data-processing workflows.
# Workflow-style example (not run on CRAN)



# This block demonstrates how .fit_standard_curve() is typically used
# inside the MFItoRAU_Adj-conversion pipeline.

# Step 1 – Prepare master file (normally from readSeroData)
master_file <- data.frame(
  Location = c("A1","A2","A3"),
  Sample   = c("S1","S2","S3"),
  Plate    = c("Plate1","Plate1","Plate1"),
  Ag1 = c(12000, 8000, 4000),
  Ag2 = c(9000,  5000, 2500)
)

# Convert antigen columns to numeric
L <- master_file |>
  dplyr::mutate(dplyr::across(-c(Location, Sample, Plate), as.numeric))
```

```
# Fake plate layout (normally from readPlateLayout)
layout <- list(Plate1 = data.frame(Location = c("A1","A2","A3"), WellType = "STD"))


# Step 2 — Load reference standard curve MFI values (dummy data)
refs <- data.frame(
  std_plate = rep("StdPlate1", 5),
  antigen   = rep("Ag1", 5),
  dilution  = c(1, 1/2, 1/4, 1/8, 1/16),
  eth_mfi   = c(14000, 7000, 3500, 1800, 900),
  png_mfi   = c(15000, 7600, 3800, 1900, 950)
)


# Step 3 — Define optimisation settings
control <- list(
  maxit  = 10000,
  abstol = 1e-8,
  reltol = 1e-6
)


# Step 4 — Fit ETH and PNG curves per standard-plate × antigen
ref_fit <- refs |>
  dplyr::group_by(.data$std_plate, .data$antigen) |>
  tidyr::nest() |>
  dplyr::mutate(
    eth_fit = purrr::map(data, ~ .fit_standard_curve(.x$eth_mfi, .x$dilution, control)),
    png_fit = purrr::map(data, ~ .fit_standard_curve(.x$png_mfi, .x$dilution, control))
  )

ref_fit
```

`.magpix_version_config` *Helper function to identify Magpix version*

Description

Helper function to identify Magpix version

Usage

```
.magpix_version_config(version)
```

Arguments

`version`	String with the raw data path.

Value

specific column names for filtering for xPONENT software v4.2 and v4.3

Author(s)

Dionne Argyropoulos

Examples

```
version = "4.2"
.magpix_version_config(version)
```

`.merge_mfitorau` *Helper function to add Sample IDs to output.*

Description

Helper function to add Sample IDs to output.

Usage

```
.merge_mfitorau(df, layout, plate_level)
```

Arguments

`df`	Data frame following 5-parameter logistic function applied.
`layout`	Output from 'readPlateLayout()'.
`plate_level`	Specific plate of interest.

Value

Processed data frame with correct Sample IDs.

Author(s)

Dionne Argyropoulos

.post_process_bioplex *Helper function to process bioplex sections*

Description

Helper function to process bioplex sections

Usage

```
.post_process_bioplex(df)
```

Arguments

df	Output from '.read_luminex_file()'

Value

List of data_raw, results, counts, blanks, stds, run

Author(s)

Dionne Argyropoulos

your_raw_data <- system.file("extdata", "example_BioPlex_plate1.xlsx", package = "SeroTrackR") df <- .read_luminex_file(your_raw_data) sections <- .post_process_bioplex(df)

.post_process_luminex *Helper function to process luminex into master_list*

Description

Helper function to process luminex into master_list

Usage

```
.post_process_luminex(sections, file_name, master_list)
```

Arguments

sections	Output from '.post_process_bioplex()'.
file_name	User input file name.
master_list	Intermediary df from 'readSeroData()'.

Value

List of data_raw, results, counts, blanks, stds, run

Author(s)

Dionne Argyropoulos

`.process_antigen_loglog` *Helper function to fit a 5-parameter logistic standard curve to dilutions*

Description

Helper function to fit a 5-parameter logistic standard curve to dilutions

Usage

```
.process_antigen_loglog(
  subset_data,
  antigen,
  dilution,
  s1_concentration,
  s_final_concentration,
  unknown_letters = c("U", "X")
)
```

Arguments

`subset_data`	Data for one plate.
`antigen`	Data for one antigen.
`dilution`	Set of five or ten.
`s1_concentration`	Concentration of highest dilution.
`s_final_concentration`	Concentration lowest dilution.
`unknown_letters`	Bioplex, Magpix or Intelliflex known unknown letters (Default = U and X).

Value

A list of the model results data frame and model.

Author(s)

Connie Li Wai Suen, Dionne Argyropoulos

.read_luminex_file *Helper function to read raw luminex files*

Description

Helper function to read raw luminex files

Usage

```
.read_luminex_file(file)
```

Arguments

file	String with the raw data path.

Value

raw data frame

Author(s)

Dionne Argyropoulos

Examples

```
your_raw_data <- system.file("extdata", "example_MAGPIX_plate1.csv", package = "SeroTrackR")
df <- .read_luminex_file(your_raw_data)
```

.relabel_columns *Relabel column names to Standardised Naming Convention*

Description

This is a helper function to be used inside ‘readSeroData()‘ to relabel columns for each plate.

Usage

```
.relabel_columns(df)
```

Arguments

df	Data frame from ‘readSeroData()‘ processing.

Value

A data fame with columns renamed

Author(s)

Dionne Argyropoulos

Examples

```
your_raw_data <- system.file("extdata", "example_MAGPIX_plate1.csv", package = "SeroTrackR")

if (
 requireNamespace("dplyr", quietly = TRUE) &&
 requireNamespace("janitor", quietly = TRUE)
) {

  # Read in raw luminex file
  df <- .read_luminex_file(your_raw_data)

  # Get the start and end rows of the data section: start = "Median", end = "Net MFI"
  row1    <- which(df$xPONENT == "Median")
  row2    <- which(df$xPONENT == "Net MFI")

  # Apply data processing pipeline, including .relabel_columns()
  df |>
    dplyr::slice((row1 + 1):(row2 - 1)) |>
    janitor::row_to_names(row_number = 1) |>
    janitor::clean_names() |>
    dplyr::select(dplyr::where(~ !all(is.na(.x)))) |>
    dplyr::filter(dplyr::if_any(dplyr::everything(), ~ !is.na(.x))) |>
    dplyr::mutate(dplyr::across(everything(), ~ gsub("NaN", 0, .))) |>
    .relabel_columns()
}
```

.setup_mfitorau_inputs

Helper function to set up MFI to RAU function

Description

Helper function to set up MFI to RAU function

Usage

```
.setup_mfitorau_inputs(df, plate_list, std_point)
```

Arguments

df	Output from 'readSeroData()'.
plate_list	Output from 'readPlateLayout()'.
std_point	Standard Point Curve: 5 = 5-point curve, 10 = 10-point curve, "PvLDH" for LDH specific curve. Default = 10. Value is an integer.

Value

A list of processed sero_data, processed plate layout, antigen names, and parameters for standard curve.

Author(s)

Dionne Argyropoulos

Examples

```
# Step 0: Load example raw data
your_raw_data <- c(
  system.file("extdata", "example_MAGPIX_plate1.csv", package = "SeroTrackR"),
  system.file("extdata", "example_MAGPIX_plate2.csv", package = "SeroTrackR")
)
your_plate_layout <- system.file(
  "extdata",
  "example_platelayout_1.xlsx",
  package = "SeroTrackR"
)

# Step 1: Read serology data and plate layout
sero_data  <- readSeroData(your_raw_data,"magpix")
plate_list <- readPlateLayout(your_plate_layout, sero_data)

# Step 2: Setup MFI to RAU
setup <- .setup_mfitorau_inputs(
  df = sero_data$results,
  plate_list = plate_list,
  std_point = 10
)
```

---

classifyResults *Random Forest Classification*

Description

This function classifies unknown samples as recently exposed or not (Note: MFItoRAU() or MFItoRAU_Adj() needs to be run first to convert to RAU).

Usage

```
classifyResults(
  mfi_to_rau_output,
  algorithm_type = "antibody_model",
```

```
  sens_spec = "balanced",
  qc_results,
  project = NULL
)
```

Arguments

mfi_to_rau_output	Output from 'MFItoRAU()' or 'MFItoRAU_Adj()'.
algorithm_type	Algorithm: "antibody_model" (PvSEM algorithm; default)
sens_spec	User-selected Sensitivity/Specificity threshold: "balanced" (default) or "90% specificity".
qc_results	Output from 'runQC()'.
project	Default = NULL. Only write "pkpfpv" if using Pk/Pf/Pv pipeline.

Value

- Data frame with exposure status for every sample. - Summary table with positive/negative results for each threshold.

Author(s)

Lauren Smith, Dionne Argyropoulos

Examples

```
# Step 0: Load example raw data
your_raw_data <- c(
  system.file("extdata", "example_MAGPIX_plate1.csv", package = "SeroTrackR"),
  system.file("extdata", "example_MAGPIX_plate2.csv", package = "SeroTrackR")
)
your_plate_layout <- system.file(
  "extdata",
  "example_platelayout_1.xlsx",
  package = "SeroTrackR"
)

# Step 1: Read serology data and plate layout
sero_data  <- readSeroData(your_raw_data,"magpix")
plate_list <- readPlateLayout(your_plate_layout, sero_data)

# Step 2: Process counts and perform quality control
qc_results  <- runQC(sero_data, plate_list)

# Step 3: Convert MFI to RAU using ETH beads
mfi_to_rau <- MFItoRAU_Adj(
  sero_data   = sero_data,
  plate_list  = plate_list,
  qc_results  = qc_results
```

```
)

# Step 4: Perform Pv classification
pv_classified <- classifyResults(
  mfi_to_rau_output = mfi_to_rau,
  algorithm_type    = "antibody_model",
  sens_spec         = "balanced",
  qc_results        = qc_results
)
```

`example_BioPlex_plate1_xlsx` *Example Serological Dataset: Bioplex Plate 1*

Description

A dataset containing raw MFI values and metadata from a sample plate run (Bioplex).

Format

A data frame with 103 rows and 15 columns.

Details

This file is stored in inst/extdata

Source

Randomised data

`example_BioPlex_plate2_xlsx` *Example Serological Dataset: Bioplex Plate 2*

Description

A dataset containing raw MFI values and metadata from a sample plate run (Bioplex).

Format

A data frame with 103 rows and 15 columns.

Source

Randomised data

`example_BioPlex_plate3_xlsx`
Example Serological Dataset: Bioplex Plate 3

Description

A dataset containing raw MFI values and metadata from a sample plate run (Bioplex).

Format

A data frame with 104 rows and 8 columns.

Details

This file is stored in inst/extdata

Source

Randomised data

`example_BioPlex_PvLDH_plate1_xlsx`
Example Serological Dataset: Bioplex PvLDH Plate 1

Description

A dataset containing raw MFI values and metadata from a sample plate run (Bioplex).

Format

A data frame with 103 rows and 15 columns.

Details

This file is stored in inst/extdata

Source

Randomised data

`example_MAGPIX_pk_10std_plate1_csv`
Example Serological Dataset: MAGPIX Plate 1 Pk Analysis 10-Point Standard Curve A dataset containing raw MFI values and metadata from a sample plate run (MAGPIX).

Description

This file is stored in inst/extdata

Format

A data frame with 614 rows and 17 columns.

Source

Randomised data

`example_MAGPIX_pk_10std_plate2_csv`
Example Serological Dataset: MAGPIX Plate 2 Pk Analysis 10-Point Standard Curve

---

Description

A dataset containing raw MFI values and metadata from a sample plate run (MAGPIX).

Format

A data frame with 614 rows and 17 columns.

Details

This file is stored in inst/extdata

Source

Randomised data

`example_MAGPIX_pk_5std_plate1_csv`
Example Serological Dataset: MAGPIX Plate 1 Pk Analysis 5-Point Standard Curve A dataset containing raw MFI values and metadata from a sample plate run (MAGPIX).

Description

This file is stored in inst/extdata

Format

A data frame with 614 rows and 17 columns.

Source

Randomised data

`example_MAGPIX_pk_5std_plate2_csv`
Example Serological Dataset: MAGPIX Plate 2 Pk Analysis 5-Point Standard Curve

---

Description

A dataset containing raw MFI values and metadata from a sample plate run (MAGPIX).

Format

A data frame with 614 rows and 17 columns.

Details

This file is stored in inst/extdata

Source

Randomised data

`example_MAGPIX_plate1_csv` *Example Serological Dataset: MAGPIX Plate 1 A dataset containing raw MFI values and metadata from a sample plate run (MAGPIX).*

Description

This file is stored in inst/extdata

Format

A data frame with 614 rows and 17 columns.

Source

Randomised data

`example_MAGPIX_plate2_csv` *Example Serological Dataset: MAGPIX Plate 2*

Description

A dataset containing raw MFI values and metadata from a sample plate run (MAGPIX).

Format

A data frame with 614 rows and 17 columns.

Details

This file is stored in inst/extdata

Source

Randomised data

example_MAGPIX_plate3_csv
Example Serological Dataset: MAGPIX Plate 3

Description

A dataset containing raw MFI values and metadata from a sample plate run (MAGPIX).

Format

A data frame with 614 rows and 17 columns.

Details

This file is stored in inst/extdata

Source

Randomised data

example_platelayout_1_xlsx
96 Well Plate Example Layout

Description

96 well plate map in a wide format used in the lab. Contains information of actual Sample ID names in each well.

Format

A data frame with 9 rows and 13 variables.

Plate Contains rows labelled "A" to "H"

1-12 Contains columns labelled "1" to "12"

Details

This file is stored in inst/extdata

`example_platelayout_pk_10std_xlsx`
10-Point Standard Curve Example Plate Layout

Description

96 well plate map in a wide format used in the lab, used for when 10-point standard curves are required for pk/pf/pv analysis. Contains information of actual Sample ID names in each well.

Format

A data frame with 9 rows and 13 variables.

Plate Contains rows labelled "A" to "H"

1-12 Contains columns labelled "1" to "12"

Details

This file is stored in inst/extdata

`example_platelayout_pk_5std_xlsx`
5-Point Standard Curve Example Plate Layout

Description

96 well plate map in a wide format used in the lab, used for when 5-point standard curves are required for pk/pf/pv analysis. Contains information of actual Sample ID names in each well.

Format

A data frame with 9 rows and 13 variables.

Plate Contains rows labelled "A" to "H"

1-12 Contains columns labelled "1" to "12"

Details

This file is stored in inst/extdata

`getAntigenCounts` *Get Count Data for each Antigen from the Raw Median Fluorescent Intensity*

Description

This function obtains the count data from the raw Median Fluorescent Intensity (MFI). This function relies on the 'readAntigens' and 'readSeroData' data processing functions.

Usage

```
getAntigenCounts(processed_counts, plate_list)
```

Arguments

`processed_counts`	Output from 'processCounts()'.
`plate_list`	Output from 'readPlateLayout()'.

Value

(i) Data frame providing bead counts per antigen per well per plate. (ii) Designates whether wells should be repeated if there are $<=$ 15 beads (repeat) or if they are sufficient with $>$ 15 beads (sufficient beads).

Author(s)

Dionne Argyropoulos

Examples

```
# Step 0: Load example raw data
your_raw_data <- c(
  system.file("extdata", "example_MAGPIX_plate1.csv", package = "SeroTrackR"),
  system.file("extdata", "example_MAGPIX_plate2.csv", package = "SeroTrackR")
)
your_plate_layout <- system.file(
  "extdata",
  "example_platelayout_1.xlsx",
  package = "SeroTrackR"
)

# Step 1: Read serology data and plate layout
sero_data  <- readSeroData(your_raw_data,"magpix")
plate_list <- readPlateLayout(your_plate_layout, sero_data)

# Step 2: Process counts and perform quality control
counts      <- processCounts(sero_data)
```

```
counts_raw  <- getCounts(counts)
sample_ids  <- getSampleID(counts, plate_list)

# Get Antigen Counts:
antigen_cts <- getAntigenCounts(counts, plate_list)
```

`getCounts` *Get Count Data from Raw Median Fluorescent Intensity*

Description

This function obtains the count data from the raw Median Fluorescent Intensity (MFI). This is an interim function used for the plotCounts function. This function relies on the ‘readAntigens‘ and ‘readSeroData‘ data processing functions.

Usage

```
getCounts(processed_counts)
```

Arguments

`processed_counts`
: Output from ‘processCounts()‘.

Value

(i) Data frame providing bead counts per well per plate. (ii) Designates whether wells should be repeated if there are $<=$ 15 beads (repeat) or if they are sufficient with > 15 beads (sufficient beads).

Author(s)

Shazia Ruybal-Pesántez, Dionne Argyropoulos

Examples

```
# Step 0: Load example raw data
your_raw_data <- c(
  system.file("extdata", "example_MAGPIX_plate1.csv", package = "SeroTrackR"),
  system.file("extdata", "example_MAGPIX_plate2.csv", package = "SeroTrackR")
)
your_plate_layout <- system.file(
  "extdata",
  "example_platelayout_1.xlsx",
  package = "SeroTrackR"
)

# Step 1: Read serology data and plate layout
```

```
sero_data  <- readSeroData(your_raw_data,"magpix")
plate_list <- readPlateLayout(your_plate_layout, sero_data)

# Step 2: Process counts and perform quality control
counts      <- processCounts(sero_data)
counts_raw  <- getCounts(counts)
```

`getCountsQC` *Get All Counts Data*

Description

This function obtains the count data from the raw Median Fluorescent Intensity (MFI). This function relies on the output of the Antigen-specific counts (‘getAntigenCounts‘) and the Well or Sample-specific counts (‘getCounts‘).

Usage

```
getCountsQC(antigen_counts_output, counts_output)
```

Arguments

`antigen_counts_output`	Output from ‘getAntigenCounts‘.
`counts_output`	Output from ‘getCounts‘.

Value

Joined data frame for all count data.

Author(s)

Dionne Argyropoulos

Examples

```
# Step 0: Load example raw data
your_raw_data <- c(
  system.file("extdata", "example_MAGPIX_plate1.csv", package = "SeroTrackR"),
  system.file("extdata", "example_MAGPIX_plate2.csv", package = "SeroTrackR")
)
your_plate_layout <- system.file(
  "extdata",
  "example_platelayout_1.xlsx",
  package = "SeroTrackR"
)
```

```
# Step 1: Read serology data and plate layout
sero_data  <- readSeroData(your_raw_data,"magpix")
plate_list <- readPlateLayout(your_plate_layout, sero_data)

# Step 2: Process counts and perform quality control
counts      <- processCounts(sero_data)
counts_raw  <- getCounts(counts)
sample_ids  <- getSampleID(counts, plate_list)
antigen_cts <- getAntigenCounts(counts, plate_list)
counts_qc   <- getCountsQC(antigen_cts, counts_raw)
```

`getGithubRelease` *Get GitHub Version*

Description

A short function to obtain the github version from the repository. This is a generalisable function that can be used for any version tags on a repo.

Usage

```
getGithubRelease(repo_owner, repo_name)
```

Arguments

`repo_owner`	GitHub Username
`repo_name`	GitHub Repository Name

Value

Version tag string

Author(s)

Dionne Argyropoulos

Examples

```
getGithubRelease(
 repo_owner = "dionnecargy",
 repo_name = "SeroTrackR"
)
```

`getPlateLayout` *Find and create a master plate layout file*

Description

Join multiple a plate layout files into one master file with multiple tabs

Usage

```
getPlateLayout(folder_path = getwd(), output_file = NULL)
```

Arguments

`folder_path`	A string containing your main folder for your project or the plate layout files. Default = current working directory.
`output_file`	A string for the path for your output master file.

Value

An .xlsx file saved to your current working directory with multiple tabs, one tab for each plate layout.

Author(s)

Dionne Argyropoulos

Examples

```
# Example 1: Create two example 96-well plates in-memory
create_plate <- function(plate_name) {
  rows <- LETTERS[1:8]
  cols <- 1:12
  df <- data.frame(plate = rows)
  for (col in cols) {
    df[[as.character(col)]] <- paste0(rows, col)
  }
  df$plate_id <- plate_name
  df
}

plate1 <- create_plate("Plate1")
plate2 <- create_plate("Plate2")

# Combine plates into a list to simulate getPlateLayout() output
master_layout <- list(
  path = tempfile(fileext = ".xlsx"),  # placeholder path
  data = list(Plate1 = plate1, Plate2 = plate2)
)
```

```
# The returned list contains:
# 1. path: the file path to the (simulated) master Excel file
# 2. data: a list of data.frames, one per plate
names(master_layout$data)  # View sheet names

# Example 2: Access individual plates directly
layout_files <- list(plate1, plate2)  # simulate individual Excel sheets

master_layout2 <- list(
  path = tempfile(fileext = ".xlsx"),  # placeholder path
  data = setNames(layout_files, c("Plate1", "Plate2"))
)

# View the resulting plate names
names(master_layout2$data)
```

---

getRepeats *Check Beads to Repeat*

Description

This function gets the count data and outputs a table of the isolates to repeat or a statement to confirm that none need to be repeated.

Usage

```
getRepeats(qc_results, plate_list)
```

Arguments

qc_results	Output from 'runQC()'.
plate_list	Output from 'readPlateLayout()'.

Value

A data frame with wells to "fail", OR if no "fail" found will return text "No repeats necessary".

Author(s)

Dionne Argyropoulos

Examples

```
# Step 0: Load example raw data and plate layout
raw_data <- c(
  system.file("extdata", "example_MAGPIX_plate1.csv", package = "SeroTrackR"),
  system.file("extdata", "example_MAGPIX_plate2.csv", package = "SeroTrackR"),
  system.file("extdata", "example_MAGPIX_plate3.csv", package = "SeroTrackR")
```

```
)
plate_layout <- system.file("extdata", "example_platelayout_1.xlsx", package = "SeroTrackR")

# Step 1: Read data and plate layout
sero_data   <- readSeroData(raw_data, platform = "magpix")
plate_list  <- readPlateLayout(plate_layout, sero_data)

# Step 2: Process counts
qc_results  <- runQC(sero_data, plate_list)

# Step 3: Identify samples to repeat
repeats_table <- getRepeats(
  qc_results = qc_results,
  plate_list = plate_list
)

# View results
repeats_table
```

`getSampleID` *Get SampleID from Plate Layout*

Description

A helper function to extract Sample ID based on plate name and row/col

Usage

```
getSampleID(processed_counts, plate_list)
```

Arguments

`processed_counts`	Output from 'processCounts()'.
`plate_list`	Plate name inside of the plate layout file.

Value

Returns the corresponding Sample ID for the correct row/column in the plate layout file. Henceforth "Sample ID" refers to the code in the plate layout file, while "Sample" is the code in the Luminex file.

Author(s)

Dionne Argyropoulos

Examples

```
# Step 0: Load example raw data
your_raw_data <- c(
  system.file("extdata", "example_MAGPIX_plate1.csv", package = "SeroTrackR"),
  system.file("extdata", "example_MAGPIX_plate2.csv", package = "SeroTrackR")
)
your_plate_layout <- system.file(
  "extdata",
  "example_platelayout_1.xlsx",
  package = "SeroTrackR"
)

# Step 1: Read serology data and plate layout
sero_data  <- readSeroData(your_raw_data,"magpix")
plate_list <- readPlateLayout(your_plate_layout, sero_data)

# Step 2: Process counts and perform quality control
counts      <- processCounts(sero_data)
counts_raw  <- getCounts(counts)
sample_ids  <- getSampleID(counts, plate_list)
```

---

makeCard *Make Card in Fluent UI*

Description

This function imports the makes a card following the Fluent UI format.

Usage

```
makeCard(title, id, content, size = 12, style = "")
```

Arguments

title	String with the large title that will be printed in the card.
id	Identifying tag for use to link.
content	A list of content to be rendered.
size	A value from 1 to 12 of the width of the screen (default = 12).
style	Value for any css styling.

Value

A "card" in the Fluent UI format with content.

Examples

```
# Minimal example creating a simple Fluent UI card.
# Safe for CRAN: runs only if shiny.fluent, htmltools, and glue are installed.

if (requireNamespace("shiny.fluent", quietly = TRUE) &&
    requireNamespace("htmltools", quietly = TRUE) &&
    requireNamespace("glue", quietly = TRUE)) {

  # Simple card content
  card_content <- list(
    htmltools::div("This is some example text inside the card.")
  )

  # Create a Fluent UI card
  makeCard(
    title   = "Example Card",
    id      = "example-card",
    content = card_content,
    size    = 6
  )
}
```

`MFItoRAU` *Median Fluorescent Intensity (MFI) to Relative Antibody Units (RAU) conversion*

Description

This function fits a 5-parameter logistic standard curve to the dilutions of the positive controls for each protein and converts the MFI values into relative antibody units (RAU) written by Connie Li Wai Suen.

Usage

```
MFItoRAU(sero_data, plate_list, qc_results, std_point = 10, project = NULL)
```

Arguments

`sero_data`	Output from ‘readSeroData()‘.
`plate_list`	Output from ‘readPlateLayout()‘.
`qc_results`	Output from ‘runQC()‘.
`std_point`	Standard Point Curve: 5 = 5-point curve, 10 = 10-point curve, "PvLDH" for LDH specific curve. Default = 10. Value is an integer.
`project`	Default = NULL. Only write "pkpfpv" if using Pk/Pf/Pv pipeline.

Value

A list of three data frames: 1. Data frame with MFI data, converted RAU data and matched SampleID's. 2. Plot information for 'plotModel' function 3. Data frame of RAU data for random forest classification use.

Author(s)

Dionne Argyropoulos, Connie Li Wai Suen

Examples

```
# Step 0: Load example raw data
your_raw_data <- c(
  system.file("extdata", "example_MAGPIX_plate1.csv", package = "SeroTrackR"),
  system.file("extdata", "example_MAGPIX_plate2.csv", package = "SeroTrackR")
)
your_plate_layout <- system.file(
  "extdata",
  "example_platelayout_1.xlsx",
  package = "SeroTrackR"
)

# Step 1: Read serology data and plate layout
sero_data  <- readSeroData(your_raw_data,"magpix")
plate_list <- readPlateLayout(your_plate_layout, sero_data)

# Step 2: Process counts and perform quality control
qc_results  <- runQC(sero_data, plate_list)

# Step 3: Convert MFI to RAU
mfi_to_rau <- MFItoRAU(
  sero_data = sero_data,
  plate_list = plate_list,
  qc_results = qc_results
)
```

MFItoRAU_Adj *Median Fluorescent Intensity (MFI) to Relative Antibody Units (RAU) conversion based on other standard*

Description

This function fits a 5-parameter logistic standard curve to the dilutions of the positive controls for each protein and converts the MFI values into relative antibody units (RAU) written by Eamon Conway.

Usage

```
MFItoRAU_Adj(sero_data, plate_list, qc_results, std_point = 10, project = NULL)
```

Arguments

sero_data	Output from 'readSeroData()'.
plate_list	Output from 'readPlateLayout()'.
qc_results	Output from 'runQC()'.
std_point	Standard Point Curve: 5 = 5-point curve, 10 = 10-point curve, "PvLDH" for LDH specific curve. Default = 10. Value is an integer.
project	Default = NULL. Only write "pkpfpv" if using Pk/Pf/Pv pipeline.

Value

A list of three data frames: 1. Data frame with MFI data, converted RAU data and matched SampleID's. 2. Plot information for 'plotModel' function. 3. Data frame of RAU data for random forest classification use.

Author(s)

Eamon Conway, Dionne Argyropoulos

Examples

```
# Step 0: Load example raw data
your_raw_data <- c(
  system.file("extdata", "example_MAGPIX_plate1.csv", package = "SeroTrackR"),
  system.file("extdata", "example_MAGPIX_plate2.csv", package = "SeroTrackR")
)
your_plate_layout <- system.file(
  "extdata",
  "example_platelayout_1.xlsx",
  package = "SeroTrackR"
)

# Step 1: Read serology data and plate layout
sero_data  <- readSeroData(your_raw_data,"magpix")
plate_list <- readPlateLayout(your_plate_layout, sero_data)

# Step 2: Process counts and perform quality control
qc_results  <- runQC(sero_data, plate_list)

# Step 3: Convert MFI to RAU using ETH beads
mfi_to_rau <- MFItoRAU_Adj(
  sero_data    = sero_data,
  plate_list   = plate_list,
  qc_results   = qc_results
)
```

MFItoRAU_LDH *Median Fluorescent Intensity (MFI) to Relative Antibody Units (RAU) conversion for LDH*

Description

This function fits a 5-parameter logistic standard curve to the dilutions of the positive controls for each protein and converts the MFI values into relative antibody units (RAU).

Usage

```
MFItoRAU_LDH(sero_data, plate_list, std_point = "PvLDH", file_path = NULL)
```

Arguments

sero_data	Output from 'readSeroData()' or 'readSeroData()'.
plate_list	Output from 'readPlateLayout()'.
std_point	Standard Point Curve: 5 = 5-point curve, 10 = 10-point curve, "PvLDH" for LDH specific curve. Default = "PvLDH".
file_path	A file path to write the .csv final file. Default: Current working directory.

Value

A data frame containing the MFI and RAU Dilution values for each sample

Author(s)

Connie Li Wai Suen, Caitlin Bourke, Dionne Argyropoulos

Examples

```
# Example demonstrating multi-plate processing workflow.
# These files are included in the SeroTrackR package under inst/extdata.

your_raw_data <- system.file("extdata", "example_BioPlex_PvLDH_plate1.xlsx", package = "SeroTrackR")

your_plate_layout <- system.file(
  "extdata", "example_platelayout_1.xlsx",
  package = "SeroTrackR"
)

# Read in raw BioPlex data
sero_data <- readSeroData(
  raw_data = your_raw_data,
  platform = "bioplex"
)
```

```
# Read matching plate layout
plate_list <- readPlateLayout(
  plate_layout = your_plate_layout,
  sero_data = sero_data
)

# Run MFI to RAU conversion
mfi_outputs <- MFItoRAU_LDH(
  sero_data = sero_data,
  plate_list = plate_list
)

# View All Outputs
mfi_outputs
```

`MFItoRAU_Pk` *Median Fluorescent Intensity (MFI) to Relative Antibody Units (RAU) conversion for Pk proteins*

Description

This function is utilised in the master function 'MFItoRAU_Plasmo()'.

Usage

```
MFItoRAU_Pk(processed_Pk, plate_list, std_point, qc_results)
```

Arguments

`processed_Pk`	df$Pk of output 'processPkPfPv()'
`plate_list`	Output of 'readPlateLayout()'
`std_point`	Standard Point Curve: 5 = 5-point curve, 10 = 10-point curve, "PvLDH" for LDH specific curve. Default = 10. Value is an integer.
`qc_results`	Output from 'runQC()'.

Value

Data frame with MFI data, converted RAU data and matched SampleID's.

Author(s)

Dionne Argyropoulos, Caitlin Bourke

`MFItoRAU_Plasmo` *Median Fluorescent Intensity (MFI) to Relative Antibody Units (RAU) conversion for Pk/Pf/Pv Master Function*

Description

This function leverages 'MFItoRAU_Pk()' and 'MFItoRAU()' to create a final MFI to RAU output for Pk/Pf/Pv analyses.

Usage

```
MFItoRAU_Plasmo(sero_data, plate_list, panel = "panel1", std_point, qc_results)
```

Arguments

`sero_data`	Output of 'readserodata_output()'
`plate_list`	Output of 'readPlateLayout()'
`panel`	Panel of Pk/Pf/Pv antigens. Default = "panel1" or user provided csv of Antigens and Species.
`std_point`	Standard Point Curve: 5 = 5-point curve, 10 = 10-point curve, "PvLDH" for LDH specific curve. Default = 10. Value is an integer.
`qc_results`	Output from 'runQC()'.

Value

A list of three data frames: 1. Data frame with MFI data, converted RAU data, matched SampleID's, all intermediate dilution conversion factors 2. Data frame with only SampleID's, MFI and RAU data 3. Data frame #2 in long-format

Author(s)

Dionne Argyropoulos, Caitlin Bourke

Examples

```
# Example demonstrating multi-plate 5-standard processing workflow.
# These files are included in the SeroTrackR package under inst/extdata.

your_raw_data_5std <- c(
  system.file("extdata", "example_MAGPIX_pk_5std_plate1.csv", package = "SeroTrackR"),
  system.file("extdata", "example_MAGPIX_pk_5std_plate2.csv", package = "SeroTrackR")
)
your_plate_layout_5std <- system.file(
  "extdata", "example_platelayout_pk_5std.xlsx",
  package = "SeroTrackR"
)
```

```
# Read in raw MAGPIX data
sero_data <- readSeroData(
  raw_data = your_raw_data_5std,
  platform = "magpix"
)

# Read matching plate layout
plate_list <- readPlateLayout(
  plate_layout = your_plate_layout_5std,
  sero_data = sero_data
)
# Quality control
qc_results  <- runQC(sero_data, plate_list)

# Run MFI to RAU conversion
mfi_outputs               <- MFItoRAU_Plasmo(
  sero_data = sero_data,
  plate_list = plate_list,
  panel = "panel1",
  std_point = 5,
  qc_results = qc_results
)

# View All Outputs
mfi_outputs
```

`plotBeadCounts` *Plot Bead Counts per Plate per Antigen*

Description

Enhances the 'plotCounts()' output by providing greater resolution, displaying antigens per plate, and enabling SampleID name visibility via hover (transformed to Plotly in server.R)

Usage

```
plotBeadCounts(qc_results)
```

Arguments

`qc_results`	Output from 'runQC()'.

Value

Dot plot with values > 15 threshold coloured in blue (sufficient beads) and less than or equal to 15 beads coloured in red (repeat) faceted by each antigen (ggplot).

Author(s)

Dionne Argyropoulos

Examples

```
# Step 0: Load example raw data
your_raw_data <- c(
  system.file("extdata", "example_MAGPIX_plate1.csv", package = "SeroTrackR"),
  system.file("extdata", "example_MAGPIX_plate2.csv", package = "SeroTrackR")
)
your_plate_layout <- system.file(
  "extdata",
  "example_platelayout_1.xlsx",
  package = "SeroTrackR"
)

# Step 1: Read serology data and plate layout
sero_data  <- readSeroData(your_raw_data,"magpix")
plate_list <- readPlateLayout(your_plate_layout, sero_data)

# Step 2: Process counts and perform quality control
qc_results  <- runQC(sero_data, plate_list)

# Step 3: Plot Bead Counts
plotBeadCounts(qc_results)
```

---

plotBlanks *Plot Raw Median Fluorescent Intensity Blanks Data*

Description

This function gets the blank sample data and plots the blank sample Median Fluorescent Intensity (MFI) values.

Usage

```
plotBlanks(sero_data, experiment_name)
```

Arguments

`sero_data`	Output from 'readSeroData()'.
`experiment_name`	User-input experiment name.

Value

Bar plot showing whether MFI values for the blanks for each antigen per plate is above or below the threshold MFI = 50 (ggplot).

Author(s)

Shazia Ruybal-Pesantez, Dionne Argyropoulos

Examples

```
# Example demonstrating how to process bead count data.
# These files are included in the SeroTrackR package under inst/extdata.

your_raw_data <- c(
  system.file("extdata", "example_MAGPIX_plate1.csv", package = "SeroTrackR"),
  system.file("extdata", "example_MAGPIX_plate2.csv", package = "SeroTrackR"),
  system.file("extdata", "example_MAGPIX_plate3.csv", package = "SeroTrackR")
)

# Read in raw MAGPIX data
sero_data <- readSeroData(
  raw_data = your_raw_data,
  platform = "magpix"
)

# Plot blanks
plotBlanks(
  sero_data = sero_data,
  experiment_name = "experiment1"
)
```

---

plotBoxPlotClassification

Plot Classification

Description

One example of data visualisation to detect the median and interquartile range of the RAU values per antigen for seropositive and seronegative individuals. Please note that the 'classifyResults()' function must be run first.

Usage

```
plotBoxPlotClassification(all_classifications, selected_threshold)
```

Arguments

`all_classifications`
: Data frame of 'classifyResults()' for all sens_spec thresholds.

`selected_threshold`
: String with the threshold.

Value

Box plots with RAU values for each protein stratified by classification (ggplot).

Author(s)

Dionne Argyropoulos

Examples

```
# Step 0: Load example raw data
your_raw_data <- c(
  system.file("extdata", "example_MAGPIX_plate1.csv", package = "SeroTrackR"),
  system.file("extdata", "example_MAGPIX_plate2.csv", package = "SeroTrackR")
)
your_plate_layout <- system.file(
  "extdata",
  "example_platelayout_1.xlsx",
  package = "SeroTrackR"
)

# Step 1: Read serology data and plate layout
sero_data  <- readSeroData(your_raw_data,"magpix")
plate_list <- readPlateLayout(your_plate_layout, sero_data)

# Step 2: Process counts and perform quality control
qc_results <- runQC(sero_data, plate_list)

# Step 3: Convert MFI to RAU using ETH beads
mfi_to_rau <- MFItoRAU_Adj(
  sero_data    = sero_data,
  plate_list   = plate_list,
  qc_results   = qc_results
)

# Step 4: Define sens/spec thresholds
sens_spec_all <- c(
  "balanced", "90% specificity"
)

# Step 5: Classify results across all thresholds
all_classifications <- purrr::map_dfr(sens_spec_all, ~{
  classifyResults(
    mfi_to_rau_output = mfi_to_rau,
    algorithm_type = "antibody_model",
    sens_spec = .x,
    qc_results = qc_results
  ) |>
  as.data.frame() |>
  dplyr::mutate(sens_spec = .x)
})
```

```
# Plot classification for a single threshold
plotBoxPlotClassification(all_classifications, "balanced")
```

`plotCounts` *Plot Bead Count Data*

Description

This function gets the count data and plots the plate image, creating a new facet (i.e., panel) for each antigen and each line represents the different plates so that they can be visualised.

Usage

```
plotCounts(qc_results, experiment_name)
```

Arguments

`qc_results`	Output from 'runQC()'.
`experiment_name`	User-input experiment name.

Value

Tile Plot showing binary result of "sufficient beads" with cut-off >15 beads and "repeat" less than or equal to 15 beads (ggplot).

Author(s)

Shazia Ruybal-Pesántez, Dionne Argyropoulos

Examples

```
# Step 0: Load example raw data
your_raw_data <- c(
  system.file("extdata", "example_MAGPIX_plate1.csv", package = "SeroTrackR"),
  system.file("extdata", "example_MAGPIX_plate2.csv", package = "SeroTrackR")
)
your_plate_layout <- system.file(
  "extdata",
  "example_platelayout_1.xlsx",
  package = "SeroTrackR"
)

# Step 1: Read serology data and plate layout
sero_data  <- readSeroData(your_raw_data,"magpix")
plate_list <- readPlateLayout(your_plate_layout, sero_data)
```

```
# Step 2: Process counts and perform quality control
qc_results  <- runQC(sero_data, plate_list)

# Step 3: Plot Counts
plotCounts(qc_results, "experiment1")
```

---

plotMFI *Median Fluorescent Intensity (MFI) Box Plots*

Description

Boxplot of the MFI values.

Usage

```
plotMFI(mfi_to_rau_output, location)
```

Arguments

mfi_to_rau_output	Output from 'MFItoRAU()' or 'MFItoRAU_Adj()' .
location	"PNG" or "ETH".

Value

Box plots with MFI values for each protein (ggplot).

Author(s)

Dionne Argyropoulos

Examples

```
# Step 0: Load example raw data
your_raw_data <- c(
  system.file("extdata", "example_MAGPIX_plate1.csv", package = "SeroTrackR"),
  system.file("extdata", "example_MAGPIX_plate2.csv", package = "SeroTrackR")
)
your_plate_layout <- system.file(
  "extdata",
  "example_platelayout_1.xlsx",
  package = "SeroTrackR"
)

# Step 1: Read serology data and plate layout
sero_data  <- readSeroData(your_raw_data,"magpix")
plate_list <- readPlateLayout(your_plate_layout, sero_data)
```

```
# Step 2: Process counts and perform quality control
qc_results <- runQC(sero_data, plate_list)

# Step 3: Convert MFI to RAU using ETH beads
mfi_to_rau <- MFItoRAU_Adj(
  sero_data   = sero_data,
  plate_list  = plate_list,
  qc_results  = qc_results
)

# Step 4: Plot MFI values
plotMFI(mfi_to_rau, "MFI")
```

`plotModel` *Plot the Median Fluorescent Intensity (MFI) to Relative Antibody Units (RAU) Results Data*

Description

This function gets the Median Fluorescent Intensity (MFI) to Relative Antibody Units (RAU) model results data and plots the model fits based on 'MFItoRAU'.

Usage

```
plotModel(mfi_to_rau_output, sero_data)
```

Arguments

`mfi_to_rau_output`	Output from 'MFItoRAU()'.
`sero_data`	Output from 'readSeroData()'.

Value

List of dot and line plots of MFI to RAU model standard curve, with each one representing an individual plate (ggplot).

Author(s)

Shazia Ruybal-Pesantez, Dionne Argyropoulos

Examples

```
# Step 0: Load example raw data
your_raw_data <- c(
  system.file("extdata", "example_MAGPIX_plate1.csv", package = "SeroTrackR"),
  system.file("extdata", "example_MAGPIX_plate2.csv", package = "SeroTrackR")
)
your_plate_layout <- system.file(
  "extdata",
  "example_platelayout_1.xlsx",
  package = "SeroTrackR"
)

# Step 1: Read serology data and plate layout
sero_data  <- readSeroData(your_raw_data,"magpix")
plate_list <- readPlateLayout(your_plate_layout, sero_data)

# Step 2: Process counts and perform quality control
qc_results <- runQC(sero_data, plate_list)

# Step 3: Convert MFI to RAU using ETH beads
mfi_to_rau <- MFItoRAU(
  sero_data = sero_data,
  plate_list = plate_list,
  qc_results = qc_results
)

# Step 4: Plot Model Results
plotModel(mfi_to_rau, sero_data)
```

---

plotModel_Adj *Plot the Median Fluorescent Intensity (MFI) to Relative Antibody Units (RAU) Results Data based on ETH standard*

Description

This function gets the Median Fluorescent Intensity (MFI) to Relative Antibody Units (RAU) model results data and plots the model fits based on 'MFItoRAU_Adj.'

Usage

```
plotModel_Adj(mfi_to_rau_output, sero_data)
```

Arguments

mfi_to_rau_output	Output from 'MFItoRAU_Adj()'.
sero_data	Output from 'readSeroData()'.

Value

List of dot and line plots of MFI to RAU model standard curve, with each one representing an individual plate (ggplot).

Author(s)

Dionne Argyropoulos

Examples

```
# Step 0: Load example raw data
your_raw_data <- c(
  system.file("extdata", "example_MAGPIX_plate1.csv", package = "SeroTrackR"),
  system.file("extdata", "example_MAGPIX_plate2.csv", package = "SeroTrackR")
)
your_plate_layout <- system.file(
  "extdata",
  "example_platelayout_1.xlsx",
  package = "SeroTrackR"
)

# Step 1: Read serology data and plate layout
sero_data  <- readSeroData(your_raw_data,"magpix")
plate_list <- readPlateLayout(your_plate_layout, sero_data)

# Step 2: Process counts and perform quality control
qc_results <- runQC(sero_data, plate_list)

# Step 3: Convert MFI to RAU using ETH beads
mfi_to_rau <- MFItoRAU_Adj(
  sero_data = sero_data,
  plate_list = plate_list,
  qc_results = qc_results
)

# Step 4: Plot Model Results
plotModel_Adj(mfi_to_rau, sero_data)
```

---

plotRAU *Relative Antibody Unit (RAU) Box Plots*

Description

Boxplot of the RAU values.

Usage

```
plotRAU(mfi_to_rau_output, location)
```

Arguments

mfi_to_rau_output	Output from ‘MFItoRAU()‘ or ‘MFItoRAU_Adj()‘ .
location	"PNG" or "ETH".

Value

Box plots with RAU values for each protein (ggplot).

Author(s)

Dionne Argyropoulos

Examples

```
# Step 0: Load example raw data
your_raw_data <- c(
  system.file("extdata", "example_MAGPIX_plate1.csv", package = "SeroTrackR"),
  system.file("extdata", "example_MAGPIX_plate2.csv", package = "SeroTrackR")
)
your_plate_layout <- system.file(
  "extdata",
  "example_platelayout_1.xlsx",
  package = "SeroTrackR"
)

# Step 1: Read serology data and plate layout
sero_data  <- readSeroData(your_raw_data,"magpix")
plate_list <- readPlateLayout(your_plate_layout, sero_data)

# Step 2: Process counts and perform quality control
qc_results <- runQC(sero_data, plate_list)

# Step 3: Convert MFI to RAU using ETH beads
mfi_to_rau <- MFItoRAU_Adj(
  sero_data   = sero_data,
  plate_list  = plate_list,
  qc_results  = qc_results
)

# Step 4: Plot RAU values
plotRAU(mfi_to_rau, "ETH")
```

plotStds *Plot Raw Median Fluorescent Intensity of Standard Curve Data*

Description

This function gets the standards data and plots the standard curves.

Usage

```
plotStds(sero_data, std_point = 10, location, experiment_name)
```

Arguments

sero_data	Output from 'readSeroData()'.
std_point	Standard Point Curve: 5 = 5-point curve, 10 = 10-point curve. Default = 10. Value is an integer.
location	"PNG" or "ETH" to filter WEHI standard curve data.
experiment_name	User-input experiment name.

Value

- Dot and line plot of standard curves (S1-S10) with PNG or Ethiopia stds underneath (ggplot). - WEHI-acceptable standard curve data on background of plot with user data.

Author(s)

Dionne Argyropoulos, Shazia Ruybal-Pesantez

Examples

```
# Example demonstrating how to process bead count data.
# These files are included in the SeroTrackR package under inst/extdata.

your_raw_data <- c(
  system.file("extdata", "example_MAGPIX_plate1.csv", package = "SeroTrackR"),
  system.file("extdata", "example_MAGPIX_plate2.csv", package = "SeroTrackR"),
  system.file("extdata", "example_MAGPIX_plate3.csv", package = "SeroTrackR")
)

# Read in raw MAGPIX data
sero_data <- readSeroData(
  raw_data = your_raw_data,
  platform = "magpix"
)

# Plot Standards
plotStds(
```

```
  sero_data = sero_data,
  location = "ETH",
  experiment_name = "experiment1"
)
```

---

plotStds_all *Plot Raw Median Fluorescent Intensity of Standard Curve Data*

Description

This function gets the standards data and plots the standard curves for any antigens (i.e., non-PvSeroTaT specific).

Usage

```
plotStds_all(sero_data, experiment_name)
```

Arguments

sero_data	Output from 'readSeroData()'.
experiment_name	User-input experiment name.

Value

- Dot and line plot of standard curves (S1-S10) - WEHI-acceptable standard curve data on background of plot with user data.

Author(s)

Shazia Ruybal-Pesantez, Dionne Argyropoulos

Examples

```
# Example demonstrating how to process bead count data.
# These files are included in the SeroTrackR package under inst/extdata.

your_raw_data <- c(
   system.file("extdata", "example_MAGPIX_plate1.csv", package = "SeroTrackR"),
   system.file("extdata", "example_MAGPIX_plate2.csv", package = "SeroTrackR"),
   system.file("extdata", "example_MAGPIX_plate3.csv", package = "SeroTrackR")
)

# Read in raw MAGPIX data
sero_data <- readSeroData(
  raw_data = your_raw_data,
```

```
  platform = "magpix"
)

# Plot Standards
plotStds_all(
  sero_data = sero_data,
  experiment_name = "experiment1"
)
```

`plotStds_PkPfPv` *Plot Raw Median Fluorescent Intensity of Pk/Pf/Pv Standard Curve Data*

Description

This function gets the standards data and plots the standard curves for antigens in the Pk/Pf/Pv panel.

Usage

```
plotStds_PkPfPv(sero_data, experiment_name, panel = "panel1")
```

Arguments

`sero_data`	Output from 'readSeroData()'.
`experiment_name`	User-input experiment name.
`panel`	Panel of Pk/Pf/Pv antigens. Default = "panel1" or user provided csv of Antigens and Species.

Value

- Dot and line plot of standard curves (S1-S10) - WEHI-acceptable standard curve data on background of plot with user data.

Author(s)

Dionne Argyropoulos

Examples

```
# Example demonstrating how to process bead count data.
# These files are included in the SeroTrackR package under inst/extdata.

your_raw_data <- c(
   system.file("extdata", "example_MAGPIX_pk_5std_plate1.csv", package = "SeroTrackR"),
   system.file("extdata", "example_MAGPIX_pk_5std_plate2.csv", package = "SeroTrackR")
)

# Read in raw MAGPIX data
sero_data <- readSeroData(
  raw_data = your_raw_data,
  platform = "magpix"
)

# Plot Standards
plotStds_PkPfPv(
  sero_data = sero_data,
  experiment_name = "experiment1",
  panel = "panel1"
)
```

`processCounts` *Process Counts from Raw Serological Data file*

Description

A helper function to process counts data.

Usage

```
processCounts(sero_data)
```

Arguments

`sero_data`	Output from 'readSeroData()'.

Value

Returns a long table of counts with "Warning" category (<15 == 1 and >= 15 == 0) for downstream wrangling.

Author(s)

Dionne Argyropoulos

Examples

```
# Example demonstrating how to process bead count data.
# These files are included in the SeroTrackR package under inst/extdata.

your_raw_data <- c(
  system.file("extdata", "example_MAGPIX_plate1.csv", package = "SeroTrackR"),
  system.file("extdata", "example_MAGPIX_plate2.csv", package = "SeroTrackR"),
  system.file("extdata", "example_MAGPIX_plate3.csv", package = "SeroTrackR")
)

# Read in raw MAGPIX data
sero_data <- readSeroData(
  raw_data = your_raw_data,
  platform = "magpix"
)

# Process counts
processed_master <- processCounts(sero_data = sero_data)
```

`processPkPfPv` *Processing Serological Data for Pk/Pf/Pv MFI to RAU conversion*

Description

This is a pre-requisite function before running the 'MFItoRAU_Plasmo()' so that the appropriate MFI to RAU conversions can be run for the respective antigens.

Usage

```
processPkPfPv(sero_data, plate_list, panel = "panel1")
```

Arguments

`sero_data`	Output of 'readSeroData()'
`plate_list`	Output of 'readPlateLayout()'
`panel`	Panel of Pk/Pf/Pv antigens. Default = "panel1" or user provided csv of Antigens and Species.

Value

A list of two data frames: 1. Data frame with Pk antigens 2. Data frame with Pf/Pv antigens

Author(s)

Dionne Argyropoulos

Examples

```
# Example demonstrating multi-plate 5-standard processing workflow.
# These files are included in the SeroTrackR package under inst/extdata.

your_raw_data_5std <- c(
  system.file("extdata", "example_MAGPIX_pk_5std_plate1.csv", package = "SeroTrackR"),
  system.file("extdata", "example_MAGPIX_pk_5std_plate2.csv", package = "SeroTrackR")
)

your_plate_layout_5std <- system.file(
  "extdata", "example_platelayout_pk_5std.xlsx",
  package = "SeroTrackR"
)

# Read in raw MAGPIX data
sero_data <- readSeroData(
  raw_data = your_raw_data_5std,
  platform = "magpix"
)

# Read matching plate layout
plate_list <- readPlateLayout(
  plate_layout = your_plate_layout_5std,
  sero_data = sero_data
)

# Process multi-species panel
processed_master <- processPkPfPv(
  sero_data = sero_data,
  plate_list = plate_list,
  panel = "panel1"
)
```

`readPlateLayout` *Read Plate Layout/s*

Description

This function imports the plate layout. Each sheet of the plate layout ".xlsx" file must contain 13 columns (labelled Plate, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12) (columns A-M) and 9 rows (Plate, A, B, C, D, E, F, G, H) (rows 1-9). *Note that the first row/column i.e., the A1 cell in excel is called "Plate". This function also checks that the plate sheet labels are consistent with the MAGPIX file input names, as a check prior to merging downstream.

Usage

```
readPlateLayout(plate_layout, sero_data)
```

Arguments

plate_layout	An ".xlsx" file with sheets labelled plate1, plate2... etc..
sero_data	Output from ‘readSeroData()‘.

Value

A list of data frames, with each one representing an individual plate.

Author(s)

Shazia Ruybal-Pesántez, Dionne Argyropoulos

Examples

```
# Example input files
your_raw_data <- c(
  system.file("extdata", "example_MAGPIX_plate1.csv", package = "SeroTrackR"),
  system.file("extdata", "example_MAGPIX_plate2.csv", package = "SeroTrackR"),
  system.file("extdata", "example_MAGPIX_plate3.csv", package = "SeroTrackR")
)

your_plate_layout <- system.file(
  "extdata",
  "example_platelayout_1.xlsx",
  package = "SeroTrackR"
)

# Step 1: Read and combine serological data
sero_data <- readSeroData(
  raw_data = your_raw_data,
  platform = "magpix"
)
# Step 2: Read plate layout
plate_list <- readPlateLayout(
  plate_layout = your_plate_layout,
  sero_data = sero_data
)
```

---

readSeroData *Read Raw Serological Data*

Description

This function imports the raw data from the Magpix or Bioplex machine and matches the sample names from the plate layout based on their plate/well location.

Usage

```
readSeroData(raw_data, platform, version = "4.2", raw_data_filenames = NULL)
```

Arguments

raw_data	String with the raw data path.
platform	"magpix", "bioplex" or "intelliflex".
version	xPONENT software version. For "magpix" can be 4.2 or 4.3. Default: 4.2.
raw_data_filenames	String with the raw data filename path. Default is NA as it can be deduced from raw_data. Needs to be a parameter for the PvSeroApp.

Value

List of data frames: (i) raw data output, (ii) cleaned all results (iii) count data, (iv) blanks only, (v) standards only, (vi) run information.

Author(s)

Dionne Argyropoulos, Shazia Ruybal-Pesantez

Examples

```
# Example raw data files (MAGPIX platform)
your_raw_data <- c(
  system.file("extdata", "example_MAGPIX_plate1.csv", package = "SeroTrackR"),
  system.file("extdata", "example_MAGPIX_plate2.csv", package = "SeroTrackR"),
  system.file("extdata", "example_MAGPIX_plate3.csv", package = "SeroTrackR")
)

# Read and combine raw serology data
sero_data <- readSeroData(
  raw_data = your_raw_data,
  platform = "magpix"
)
```

---

renderClassificationTable

All Classification Data

Description

This function runs the classification algorithm for all possible sensitivity and specificity options.

Usage

```
renderClassificationTable(mfi_to_rau_output, algorithm_type, qc_results)
```

Arguments

mfi_to_rau_output	Output from ‘MFItoRAU()‘ or ‘MFItoRAU_Adj()‘.
algorithm_type	Algorithm: "antibody_model" (PvSEM algorithm; default)
qc_results	Output from ‘runQC()‘.

Value

A table of all classification outputs.

Author(s)

Dionne Argyropoulos

Examples

```
# Step 0: Load in Raw Data
your_raw_data <- c(
  system.file("extdata", "example_MAGPIX_plate1.csv", package = "SeroTrackR"),
  system.file("extdata", "example_MAGPIX_plate2.csv", package = "SeroTrackR")
)
your_plate_layout <- system.file("extdata", "example_platelayout_1.xlsx", package = "SeroTrackR")

# Step 1: Reading in Raw Data
sero_data     <- readSeroData(raw_data = your_raw_data, "magpix")
plate_list    <- readPlateLayout(
  plate_layout = your_plate_layout,
  sero_data = sero_data
)

# Step 2: Quality Control and MFI to RAU
qc_results <- runQC(sero_data, plate_list)

# Step 4: Run MFI to RAU (e.g., using ETH beads)
mfi_to_rau_output  <- MFItoRAU_Adj(sero_data, plate_list, qc_results)

# Step 5: Render classification table
renderClassificationTable(
  mfi_to_rau_output = mfi_to_rau_output,
  algorithm_type = "antibody_model",
  qc_results = qc_results
)
```

renderDetailsList *Create a Fluent UI Table*

Description

This function makes the table in a Fluent UI format.

Usage

```
renderDetailsList(df)
```

Arguments

df	Any processed data frame

Value

A table in the Fluent UI format

Author(s)

Dionne Argyropoulos

Examples

```
# Minimal example using a small data frame.
# This example is safe for CRAN because it runs only if
# shiny.fluent and htmltools are installed.

if (requireNamespace("shiny.fluent", quietly = TRUE) &&
    requireNamespace("htmltools", quietly = TRUE)) {

  # Tiny example data frame
  example_df <- data.frame(
    Sample = c("A", "B"),
    Value = c(10, 20),
    stringsAsFactors = FALSE
  )

  # Render Fluent UI DetailsList
  renderDetailsList(example_df)
}
```

`renderQCReport` *Generate QC PDF Report*

Description

Generate QC PDF Report

Usage

```
renderQCReport(
  raw_data,
  plate_layout,
  platform,
  experiment_name = "experiment1",
  date = format(Sys.Date(), "%Y%m%d"),
  experiment_notes = "no notes",
  location,
  std_point = 10,
  path = "."
)
```

Arguments

`raw_data`	A string with the raw data path.
`plate_layout`	A string with the plate layout path.
`platform`	A string: "magpix", "intelliflex", or "bioplex".
`experiment_name`	A string for experiment name.
`date`	A string or Date. Defaults to today's date.
`experiment_notes`	A string of notes. Default is "no notes".
`location`	A string for experiment location: "ETH" or "PNG" accepted.
`std_point`	Standard Point Curve: 5 = 5-point curve, 10 = 10-point curve, "PvLDH" for LDH specific curve. Default = 10. Value is an integer.
`path`	Output path for the PDF file. Defaults to current working directory.

Value

Rendered PDF report.

Author(s)

Dionne Argyropoulos

Examples

```
## Not run on CRAN because it requires interactive rendering and can be slow:
## Not run:
  # Example raw data files (MAGPIX platform)
  your_raw_data <- c(
    system.file("extdata", "example_MAGPIX_plate1.csv", package = "SeroTrackR"),
    system.file("extdata", "example_MAGPIX_plate2.csv", package = "SeroTrackR"),
    system.file("extdata", "example_MAGPIX_plate3.csv", package = "SeroTrackR")
  )

  # Example plate layout file
  your_plate_layout <- system.file(
    "extdata",
    "example_platelayout_1.xlsx",
    package = "SeroTrackR"
  )

  # Generate the QC PDF report
  renderQCReport(
    raw_data     = your_raw_data,
    plate_layout = your_plate_layout,
    platform     = "magpix",
    location     = "ETH"
  )

## End(Not run)
```

`renderReport` *Render Markdown report*

Description

A short function to render the rmarkdown report on Shiny.

Usage

```
renderReport(input, output, params)
```

Arguments

`input`	Input files
`output`	Output files
`params`	Parameters to generate outputs.

Value

PDF output.

Author(s)

Dionne Argyropoulos

Examples

```
# Minimal example that renders a temporary Rmd file.
# Safe for CRAN because it only writes to tempdir()
## Not run:
if (requireNamespace("rmarkdown", quietly = TRUE) &&
    rmarkdown::pandoc_available()) {

  # Create a temporary Rmd that declares params in the YAML
  rmd_file <- tempfile(fileext = ".Rmd")
  writeLines(c(
    "---",
    "title: \"Test Report\"",
    "output: html_document",
    "params:",
    "  value: 0",
    "---",
    "",
    "This is a test report.",
    "",
    "Parameter value: `r params$value`"
  ), con = rmd_file)

  # Output location
  out_file <- tempfile(fileext = ".html")

  # Example parameters to pass in
  example_params <- list(value = 123)

  # Render report
  renderReport(
    input  = rmd_file,
    output = out_file,
    params = example_params
  )

  # Optionally inspect the output path
  out_file
}

## End(Not run)
```

`renderTwoCols` *Create two columns in Fluent UI*

Description

This function creates two columns in the Fluent UI format.

Usage

```
renderTwoCols(
  first_col,
  second_col,
  first_width = "50%",
  second_width = "50%"
)
```

Arguments

first_col	A list of content for the first column.
second_col	A list of content for the second column.
first_width	Percent width of the column space (default: 50%).
second_width	Percent width of the column space (default: 50%).

Value

Fluent UI window with two columns.

Author(s)

Dionne Argyropoulos

Examples

```
# Minimal example using htmltools elements.
# This example runs without starting a Shiny app and is safe for CRAN.

if (requireNamespace("shiny.fluent", quietly = TRUE) &&
    requireNamespace("htmltools", quietly = TRUE)) {

  # Create simple content for each column
  col1 <- list(htmltools::div("First column content"))
  col2 <- list(htmltools::div("Second column content"))

  # Render two columns with default widths
  renderTwoCols(first_col = col1, second_col = col2)
}
```

---

runLDHPipeline *Run LDH Pipeline from Start to End*

Description

A master function combining the entire LDH pipeline into one command to run in R.

Usage

```
runLDHPipeline(
  raw_data,
  plate_layout,
  platform = "bioplex",
 dilution = c(1e+06, 333333.33, 111111.11, 37037.04, 12345.68, 4115.23, 1371.74, 457.25,
    152.42, 50.81),
  experiment_name = "experiment1",
  file_path = NULL
)
```

Arguments

raw_data	String with the raw data path.
plate_layout	An ".xlsx" file with sheets labelled plate1, plate2... etc.
platform	"magpix" or "bioplex". Default: "Bioplex"
dilution	A list of numbers ranging from S1 to S10. Default: 1000000, 333333.33, 111111.11, 37037.04, 12345.68, 4115.23, 1371.74, 457.25, 152.42, 50.81.
experiment_name	User-input experiment name. Default: "experiment1".
file_path	A file path to write the .csv final file. Default: Current working directory.

Value

A data frame containing the MFI and RAU Dilution values for each sample, QC plots for standard curve, bead counts and blanks.

Author(s)

Dionne Argyropoulos

Examples

```
# Example input files
your_raw_data <- system.file(
  "extdata",
  "example_BioPlex_PvLDH_plate1.xlsx",
  package = "SeroTrackR"
)
your_plate_layout <- system.file(
  "extdata",
  "example_platelayout_1.xlsx",
  package = "SeroTrackR"
)

# Run full LDH processing pipeline
runLDHPipeline(
  raw_data      = your_raw_data,        # Vector of raw data files
```

```
  plate_layout  = your_plate_layout,    # Plate layout file
)
```

`runPlasmoPipeline` *Run Pk/Pf/Pv Data Analysis Pipeline from Start to End*

Description

Run Pk/Pf/Pv Data Analysis Pipeline from Start to End

Usage

```
runPlasmoPipeline(
  raw_data,
  platform = "magpix",
  plate_layout,
  panel = "panel1",
  std_point,
  experiment_name = "experiment1",
  classify = "Yes",
  algorithm_type = "antibody_model",
  sens_spec = "balanced"
)
```

Arguments

`raw_data`	String with the raw data path.
`platform`	"magpix" or "bioplex". Default: "Bioplex"
`plate_layout`	An ".xlsx" file with sheets labelled plate1, plate2... etc.
`panel`	Panel of Pk/Pf/Pv antigens. Default = "panel1" or user provided csv of Antigens and Species.
`std_point`	Standard Point Curve: 5 = 5-point curve, 10 = 10-point curve. Value is an integer.
`experiment_name`	User-input experiment name. Default: "experiment1".
`classify`	"Yes" or "No" depending on whether you would like classification or not. Default = "Yes".
`algorithm_type`	Algorithm: "antibody_model" (PvSEM algorithm; default)
`sens_spec`	User-selected Sensitivity/Specificity threshold: "balanced" (default) or "90% specificity".

Value

A data frame containing the MFI and RAU Dilution values for each sample, QC plots for standard curve, bead counts and blanks.

Author(s)

Dionne Argyropoulos

Examples

```
# Helper to avoid repetition in examples
run_example_std <- function(std_point) {
  # Load raw data for given standard curve
  your_raw_data <- c(
    system.file("extdata",
                paste0("example_MAGPIX_pk_", std_point, "std_plate1.csv"),
                package = "SeroTrackR"),
    system.file("extdata",
                paste0("example_MAGPIX_pk_", std_point, "std_plate2.csv"),
                package = "SeroTrackR")
  )

  layout_file <- system.file(
    "extdata",
    paste0("example_platelayout_pk_", std_point, "std.xlsx"),
    package = "SeroTrackR"
  )

  # Run pipeline
  runPlasmoPipeline(
    raw_data = your_raw_data,
    platform = "magpix",
    plate_layout = layout_file,
    panel = "panel1",
    std_point = std_point,
    experiment_name = paste0(std_point, "-point standard curve")
  )
}

# ---- 5-point standard curve ----
results_5std <- run_example_std(5)

# ---- 10-point standard curve ----
results_10std <- run_example_std(10)
```

`runPvSeroPipeline` *Run PvSero Pipeline from Start to End*

Description

A master function combining the entire PvSeroApp pipeline into one command to run in R.

Usage

```
runPvSeroPipeline(
  raw_data,
  plate_layout,
  platform = "magpix",
  location,
  experiment_name = "experiment1",
  std_point = 10,
  classify = "Yes",
  algorithm_type = "antibody_model",
  sens_spec = "balanced"
)
```

Arguments

raw_data	String with the raw data path.
plate_layout	An ".xlsx" file with sheets labelled plate1, plate2... etc.
platform	"magpix" or "bioplex". Default = "magpix".
location	"PNG" or "ETH" to filter WEHI standard curve data.
experiment_name	User-input experiment name.
std_point	Standard Point Curve: 5 = 5-point curve, 10 = 10-point curve, "PvLDH" for LDH specific curve. Default = 10. Value is an integer.
classify	"Yes" or "No" depending on whether you would like classification or not. Default = "Yes".
algorithm_type	Algorithm: "antibody_model" (PvSEM algorithm; default)
sens_spec	User-selected Sensitivity/Specificity threshold: "balanced" (default) or "90% specificity".

Value

classifyResults_output, stdcurve_plot, plateqc_plot, check_repeats_output, blanks_plot, model_plot

Author(s)

Dionne Argyropoulos

Examples

```
# Example data supplied with the package
your_raw_data <- c(
  system.file("extdata", "example_MAGPIX_plate1.csv", package = "SeroTrackR"),
  system.file("extdata", "example_MAGPIX_plate2.csv", package = "SeroTrackR"),
  system.file("extdata", "example_MAGPIX_plate3.csv", package = "SeroTrackR")
)
```

```
plate_layout <- system.file(
  "extdata", "example_platelayout_1.xlsx", package = "SeroTrackR"
)

# Run full pipeline including classification
runPvSeroPipeline(
  raw_data = your_raw_data,
  plate_layout = plate_layout,
  platform = "magpix",
  location = "PNG",
  experiment_name = "experiment1",
  std_point = 10,
  algorithm_type = "antibody_model",
  sens_spec = "balanced",
  classify = "Yes"
)

# Run processing pipeline only (no classification)
runPvSeroPipeline(
  raw_data = your_raw_data,
  plate_layout = plate_layout,
  platform = "magpix",
  location = "PNG",
  experiment_name = "experiment1",
  std_point = 10,
  algorithm_type = "antibody_model",
  sens_spec = "balanced",
  classify = "No"
)
```

`runQC` *Run Quality Control Pipeline*

Description

A master function containing each quality control processing step.

Usage

```
runQC(sero_data, plate_list)
```

Arguments

`sero_data`	Output from ‘readSeroData()‘.
`plate_list`	Output from ‘readPlateLayout()‘.

Value

processCounts_output, getCounts_output, sampleid_output, getAntigenCounts_output, getCountsQC_output

Author(s)

Dionne Argyropoulos

Examples

```
# Example data supplied with the package
your_raw_data <- c(
  system.file("extdata", "example_MAGPIX_plate1.csv", package = "SeroTrackR"),
  system.file("extdata", "example_MAGPIX_plate2.csv", package = "SeroTrackR"),
  system.file("extdata", "example_MAGPIX_plate3.csv", package = "SeroTrackR")
)

your_plate_layout <- system.file(
  "extdata", "example_platelayout_1.xlsx", package = "SeroTrackR"
)

# Read serology data and plate layout
sero_data  <- readSeroData(your_raw_data,"magpix")
plate_list <- readPlateLayout(your_plate_layout, sero_data)

# Run full pipeline including classification
runQC(
  sero_data = sero_data,
  plate_list = plate_list
)
```

Index