Package 'dpcR'

October 7, 2014

Type Package

Title Digital PCR Analysis

Version 0.1.3.1

LazyData true

Date 2014-10-02

Description Analysis, visualisation and simulation of digital PCR experiments.

License GPL-3

URL https://github.com/michbur/dpcR

BugReports https://github.com/michbur/dpcR/issues

Depends R (>= 3.0.0), methods

Imports binom, chipPCR, e1071, dgof, multcomp, qpcR, pracma,rateratio.test, signal, shiny, spatstat

Suggests knitr

VignetteBuilder knitr

NeedsCompilation no

Repository CRAN

Collate 'AUCtest.R' 'adpcr-class.R' 'adpcr2ppp.R' 'ddpcr-class.R'
'bind_dpcr.R' 'bioamp.R' 'calc_breaks.R' 'calc_lambda.R'
'compare_dens.R' 'count_test-class.R' 'create_dpcr.R'
'dpcR-package.R' 'dpcr_calculator.R' 'dpcr_density.R'
'dpcr_density_gui.R' 'extract_dpcr.R' 'fit_adpcr.R' 'fl.R'
'get_k_n.R' 'limit_cq.R' 'many_peaks.R' 'modlist.R' 'moments.R'
'num2int.R' 'pds.R' 'pds_raw.R' 'plot_distr.R' 'plot_panel.R'
'plot_vf_circ.R' 'plot_vic_fam.R' 'print_summary.R'
'qpcr_analyser.R' 'qpcrpp.R' 'rtadpcr.R' 'safe_efficiency.R'
'show_dpcr.R' 'sim_adpcr.R' 'sim_ddpcr.R' 'simulations.R'
'summary_dpcr.R' 'test_counts.R' 'test_panel.R' 'test_peaks.R'
'test_ratio.R' 'valid_amp.R' 'y_val_conf.R'

Author Michal Burdukiewicz [cre, aut],Stefan Roediger [aut]

Maintainer Michal Burdukiewicz <michalburdukiewicz@gmail.com>

Date/Publication 2014-10-07 09:39:31

R topics documented:

dpcR-package *Digital PCR Analysis*

Description

The dpcR package is a collection of functions for a digital Polymerase Chain Reaction (dPCR) analysis. dPCR comprises methods to quantify nucleic acids, copy number variations (CNV), homo-/heterozygosity, and rare mutations (including single nucleotide polymorphisms (SNP)). The chemical basis of dPCR is similar to conventional PCR but the reaction-mix is divided into hundredths to thousands of small compartments with parallel amplifications reactions. The analysis is based on counting the number of positive compartments and to relate it to the total number of compartments be means of Poission statistics which enables an absolute quantification. The package includes plot functions, summary functions, data sets and simulations for dPCR and customizable GUI creators for droplet digital PCRs and chamber-based digital PCRs. The authors of the package aim to include all statistical approaches published in peer-review literature and additional selected sources of expertise currently available and to make them available to the scientific community in an open and cross-platform environment. As such the dpcR packages has a list of expressions/functions and may serve in future a reference to a unified nomenclature in dpcR. The package is primarily targeted at researchers who which to use it with an existing technology or during the development of novel digital PCR systems. In addition the dpcR package provides interactive tools that can be used in classes or by individuals to better learn about digital PCR concepts and data interpretation.

Details

Package:	dpcR
Type:	Package
Version:	0.1.1
Date:	2013-09-07
License:	GPL2

Author(s)

Michal Burdukiewicz, Stefan Roediger.

Maintainer: Michal Burdukiewicz <michalburdukiewicz@gmail.com>

References

Huggett J, Foy CA, Benes V, Emslie K, Garson JA, Haynes R, Hellemans J, Kubista M, Mueller RD, Nolan T, Pfaffl MW, Shipley GL, Vandesompele J, Wittwer CT, Bustin SA *The Digital MIQE Guidelines: Minimum Information for Publication of Quantitative Digital PCR Experiments* Clinical Chemistry, 2013. 59(6): p.892-902.

Vogelstein B, Kinzler KW, *Digital PCR*. PNAS, 1999. 96(16): p. 9236-9241.

See Also

qpcR.news.

Examples

```
adpcr <- sim_adpcr(m = 400, n = 765, times = 20, pos_sums = FALSE, n_panels = 1)
plot_panel(adpcr, 45, 17, col = "green")
pos_chambers <- sum(adpcr > 0)
dpcr_density(k = pos_chambers, n = 765)
```

---

adpcr-class *Class* "adpcr" - *end-point array digital PCR experiments*

Description

A class specifically designed to contain results from end-point array digital PCR experiments. Data is represented as matrix, where each column describes different experiment. Type of data in all columns is specified in slot "type" and could be a number of molecules "nm", a number of positive droplets "tnm" (in this case whole experiment is represented by one row), a cycle threshold of each well "ct" or fluorescence values "fluo".

Slots

list(".Data") "matrix" containing data from array. See Description.

list("n") Object of class "integer" equal to the number of partitions in each experiment.

list("breaks") "numeric" vector giving the number of intervals into which .Data should be cut.

list("type") Object of class "character" defining type of data. Could be "nm" (number of molecules per partition), "tp" (total number of positive wells in panel), "fluo" (fluorescence) or "ct" (threshold cycle).

Author(s)

Michal Burdukiewicz.

See Also

Ploting and management: bind_dpcr, extract_dpcr, plot_panel.

Simulation: sim_adpcr.

Real-time array digital PCR: rtadpcr.

Droplet digital PCR: ddpcr.

Examples

```
rand_array <- sim_adpcr(400, 1600, 100, pos_sums = FALSE, n_panels = 5)
one_rand_array <- extract_dpcr(rand_array, 1)
plot_panel(one_rand_array, 40, 40)
```

adpcr2ppp *Convert adpcr to ppp*

Description

Quick conversion of adpcr object to the list of ppp.objects.

Usage

```
adpcr2ppp(input, nx_a, ny_a)
```

Arguments

input	Object of the adpcr class containing data from one or more panels.
nx_a	Number of columns in a plate.
ny_a	Number of rows in a plate.

Details

Each plate is independently converted by ppp function. marks attached to each point represent values contained by the adpcr object.

Value

A list containing objects with class ppp.object with the length equal to the number of plates (minimum 1).

Author(s)

Michal Burdukiewcz, Stefan Roediger.

See Also

ppp.object, ppp.

Examples

```
many_panels <- sim_adpcr(m = 400, n = 765, times = 1000, pos_sums = FALSE,
                  n_panels = 5)

# Convert all plates to ppp objects
adpcr2ppp(many_panels, nx_a = 45, ny_a = 17)

# Convert all plates to ppp objects and get third plate
third_plate <- adpcr2ppp(many_panels, nx_a = 45, ny_a = 17)[[3]]

# Convert only third plate to ppp object
third_plate2 <- adpcr2ppp(extract_dpcr(many_panels, 3), nx_a = 45, ny_a =
```

```
17)

# Check the class of a new object
class(third_plate2)

# It's a list with the length 1. The third plate is a first element on this
list
class(third_plate2[[1]])
```

`bind_dpcr-methods` *Bind dpcr objects*

Description

A convinient wrapper around `cbind` and `rbind` tailored specially for binding multiple objects containing results from digital PCR experiments.

Usage

```
bind_dpcr(input, ...)
```

Arguments

`input`	an object of class `adpcr` or `ddpcr` or a list.
`...`	objects of class `adpcr` or `ddpcr`. See Details. If `input` is a list, ignored.

Details

In case of `adpcr` or `ddpcr` objects, `bind_dpcr` can works analogously to `cbind`), but without recycling. In case on unequal length, shorter objects will be filled in with additional `NA` values. The original length is always preserved in `n` slot.

`bind_dpcr` automatically names binded experiments using format `x.y`, where `x` is number of object passed to function and `y` is a number of experiment in a given object.

Value

An object of class `adpcr` or `ddpcr`, depending on the input.

Note

Because `bind_dpcr` calls `do.call` function, binding together at the same time more than 500 objects can lead to 'stack overflow' error.

Author(s)

Michal Burdukiewicz

See Also

Opposite function: extract_dpcr

Examples

```
bigger_array <- sim_adpcr(400, 765, 1000, pos_sums = FALSE, n_panels = 5)
smaller_array <- sim_adpcr(100, 700, 1000, pos_sums = FALSE, n_panels = 3)
bound_arrays <- bind_dpcr(bigger_array, smaller_array)

smaller_droplet <- sim_ddpcr(m = 7, n = 20, times = 5, n_exp = 2)
bigger_droplet <- sim_ddpcr(m = 15, n = 25, times = 5, n_exp = 4)
biggest_droplet <- sim_ddpcr(m = 15, n = 35, times = 5, n_exp = 1)
bound_droplets <- bind_dpcr(smaller_droplet, bigger_droplet, biggest_droplet)
```

---

bioamp *A function to analyze plot the raw data from a BioRad droplet digital PCR experiment*

Description

bioamp is a function to plot and analyze the amplitude data of a BioRad droplet digital PCR experiment.

Usage

```
bioamp(data = data, amp_x = 1, amp_y = 2, cluster = 3, robust = TRUE,
  plot = TRUE, stat = TRUE, xlab = "Assay 1 Amplitude",
  ylab = "Assay 2 Amplitude", ...)
```

Arguments

data	object of class what containing the amplitude data.
amp_x	is the first amplitude (x-axis).
amp_y	is the second amplitude (y-axis).
cluster	are the clusters of the plot.
robust	Is the method used to calculate the location (mean or median) and dispersion (standard deviation or median absolute deviation).
plot	logical, if TRUE, the plot is printed.
stat	logical, if TRUE, the statistics of the droplet digital PCR experiment are calculated.
xlab	x-label of the plot.
ylab	y-label of the plot.
...	other arguments passed to the plot function (see plot.default for details).

Author(s)

Stefan Roediger, Michal Burdukiewcz

Examples

```
par(mfrow = c(1,2))
bioamp(data = pds_raw[["A01"]], main = "Well A01", pch = 19)
bioamp(data = pds_raw[["A02"]], main = "Well A02", pch = 19)
par(mfrow = c(1,1))
```

---

compare_dens *Plot and Compare Densities*

Description

Plot empirical and theoretical density of the result of the digital PCR experiment.

Usage

```
compare_dens(input, moments = TRUE, ...)
```

Arguments

input	object of class adpcr or ddpcr containing only one panel.
moments	logical, if TRUE, both theoretical and empirical moments are printed on the plot.
...	other arguments passed to the plot function.

Author(s)

Michal Burdukiewcz.

See Also

moments is used to calculate moments of Poisson distribution.

Examples

```
adpcr_big <- sim_adpcr(m = 35, n = 40, times = 50, pos_sums = FALSE, n_panels = 1)
compare_dens(adpcr_big, moments = TRUE)
```

count_test *Class* "count_test"

Description

A class for results of test_counts function.

Usage

```
## S4 method for signature 'count_test'
summary(object)

## S4 method for signature 'count_test'
show(object)

## S4 method for signature 'count_test'
plot(x, aggregate = FALSE)
```

Arguments

object	of class count_test.
x	object of class count_test.
aggregate	logical, if TRUE experiments are aggregated according to their group.

Details

In case of aggregated plot, mean confidence intervals for groups are presented as dashed lines.

Methods (by generic)

- summary: Summary statistics of assigned groups.
- show: Print both group_coef and t_res
- plot:

Slots

group_coef "data.frame" containing experiments, groups to which they belong and calculated values of rate.

t_res "matrix" containing result of t-test.

Author(s)

Michal Burdukiewicz.

See Also

Nothing yet.

`create_dpcr` *Create dpcR object*

Description

Quick creation of `adpcr` and`ddpcr` objects.

Usage

```
create_dpcr(data, n, threshold = NULL, breaks = NULL, type, adpcr = TRUE)
```

Arguments

`data`	a `"numeric"` vector or matrix of data from dPCR experiments. Data frames will be converted to matrices.
`n`	`"integer"` equal to number of partitions.
`threshold`	`"numeric"` value giving the threshold above which droplet is counted as positive. Ignored if `adpcr` is `TRUE`.
`breaks`	`"numeric"` vector giving the number of intervals into which `data` should be cut. Ignored if `adpcr` is `FALSE`.
`type`	Object of class `"character"` defining type of data. Could be `"nm"` (number of molecules per partition), `"tp"` (total number of positive wells in the panel), `"fluo"` (fluorescence) or `"ct"` (threshold cycle).
`adpcr`	logical. If `TRUE`, function creates `adpcr` object. If `FALSE`, function creates `ddpcr` object.

Details

This function assists in creation of objects used by other functions of the package. Its inbuilt capabilities include checking the correctness of arguments.

A warning is prompted whenever any of arguments is converted to other type.

Value

An `adpcr` or `ddpcr` object.

Note

Currently only end-point measurements are supported.

Author(s)

Michal Burdukiewicz, Stefan Roediger.

Examples

```
# Droplet digital PCR example
sample_runs <- matrix(rpois(60, lambda = 1.5), ncol = 2)
ddpcr1 <- create_dpcr(sample_runs[,1], n = 30L,
threshold = 1, type = "nm", adpcr = FALSE)
ddpcr2 <- create_dpcr(sample_runs[,2], n = 30L,
threshold = 1, type = "nm", adpcr = FALSE)
plot_vic_fam(ddpcr1, ddpcr2)

# Array digital PCR example
sample_adpcr <- create_dpcr(rpois(765, lambda = 0.8), n = 765L,
    type = "nm", adpcr = TRUE)
plot_panel(sample_adpcr, 45, 17)
```

---

ddpcr-class *Class* "ddpcr"

Description

A class specifically designed to contain results from droplet digital PCR experiments. Data is represented as matrix, where each column describes different experiment. Type of data in all columns is specified in slot "type" and could be a by number of molecules "nm", number of positive droplets "tnm" (in this case whole experiment is represented by one row) or fluorescence "fluo".

Slots

list(".Data") "matrix" containing data from all droplets. See Description.

: "matrix" containing data from all droplets. See Description.

list("n") "integer" representing number of partitions.

: "integer" representing number of partitions.

list("threshold") "numeric" value giving the threshold above which droplet is counted as positive.

: "numeric" value giving the threshold above which droplet is counted as positive.

list("type") Object of class "character" defining type of data. Could be "nm" (Number of molecules per partition), "tp" (number of positive droplets) or "fluo" (fluorescence).

: Object of class "character" defining type of data. Could be "nm" (Number of molecules per partition), "tp" (number of positive droplets) or "fluo" (fluorescence).

Author(s)

Michal Burdukiewicz.

See Also

Ploting and managment: bind_dpcr, extract_dpcr, plot_vic_fam.

Simulation: sim_ddpcr.

Array digital PCR: adpcr.

Examples

```
ddpcr_fluo <- sim_ddpcr(m = 10, n = 20, times = 5, fluo = list(0.1, 0))
plot(ddpcr_fluo)

ddpcr <- sim_ddpcr(m = 10, n = 20, times = 5)
```

---

dpcr_density *Calculate Density of Digital PCR*

Description

A function which calculates and plots the density of the number of positive molecules or the average number of molecules per partition. Can be used for both array digital PCR and droplet digital PCR.

Usage

```
dpcr_density(k, n, average = FALSE, methods = "wilson", conf.level = 0.95,
  plot = TRUE)
```

Arguments

k	Total number of positive molecules.
n	Total number of partitions.
average	If TRUE, calculates density of the average number of molecules per partition. If FALSE, instead performs calculations for the total number of positive molecules.
methods	Which method of calculating confidence interval should be used. Possible values are: "wilson", "agresti-coull", "exact", "prop.test", "profile", "lrt", "asymptotic", "bayes", "cloglog", "logit", "probit". Default value is "wilson". See Details.
conf.level	The level of confidence to be used in the confidence interval. Values from 0 to 1 and -1 to 0 are acceptable.
plot	If TRUE, plots density plot.

Details

Confidence interval is calculated by binom.confint.

Value

A data frame with one row containing bounds of the confidence intervals and a name of the method used to calculate them.

Note

The browser-based graphical user interface for this function is accessible as dpcr_density_gui.

Author(s)

Michal Burdukiewicz, Stefan Roediger.

References

Brown, Lawrence D., T. Tony Cai, and Anirban DasGupta. *Confidence Intervals for a Binomial Proportion and Asymptotic Expansions.* The Annals of Statistics 30, no. 1 (February 2002): 160–201.

See Also

binom.confint, dpcr_density_gui.

Examples

```
# Calculate the average number of molecules per partition and show the area
# of the confidence interval (left plot) and the area within the
# confidence interval
par(mfrow = c(1,2))
dpcr_density(k = 25, n = 55, average = TRUE, methods = "wilson",
     conf.level = 0.9)
dpcr_density(k = 25, n = 55, average = TRUE, methods = "wilson",
     conf.level = -0.9)

# By setting average to FALSE the total number of positive molecules is
# calculated
par(mfrow = c(1,1))
dpcr_density(k = 25, n = 55, average = FALSE, methods = "wilson",
     conf.level = 0.95)

# Example of an artificial chamber dPCR experiment using the reps384 data
# set from qpcR. The function cpD2limiter is used to calculate the cpD2
# value and converts all values between a defined range to 1 and the
# remaining to 0.
cpD2limiter <- function(data = data, cyc = 1, fluo.range = c(NA),
                        Cq.range = c(NA, NA)) {
  cpD2 <- vector()
  cpD2.res <- vector()
  pb <- txtProgressBar(min = 1, max = length(fluo.range), initial = 0,
style = 3)
  for (i in fluo.range) {
    cpD2.tmp <- efficiency(pcrfit(data = data, cyc = cyc, fluo = i,
   model = l5), plot = FALSE)$cpD2
    cpD2 <- c(cpD2, cpD2.tmp)
    if (Cq.range[1] <= cpD2.tmp && cpD2.tmp <= Cq.range[2]) {
    cpD2.res.tmp <- 1
    }
    else(cpD2.res.tmp <- 0)
    cpD2.res <- c(cpD2.res, cpD2.res.tmp)
    setTxtProgressBar(pb, i)
    }
  close(pb)
```

```
  out <- cbind(cpD2, cpD2.res)
  colnames(out) <- c("cpD2", "result")
  return(out)
  }

# Cq.range defines a range to convert cpD2 values into positive (1)
# and negative (0) chambers.The dataset reps384 is used as sample.
# results.dPCR contains a column with the cpD2 values and a column with
# converted values.
## Not run:
Cq.range <- c(18.1, 18.3)
results.dPCR <- cpD2limiter(data = reps384, cyc = 1,
fluo.range = c(2L:ncol(reps384)), Cq.range = Cq.range)

# Get the number of positive reactions k.tmp and the total number of
# reactions n.tmp.
k.tmp <- sum(results.dPCR[, 2]) # 191
n.tmp <- nrow(results.dPCR) # 379

# Generate an amplification plot from the reps384 data set along with the
# density of the number of positive molecules or the average number of
# molecules per partition.
par(mfrow = c(1,2))
plot(NA, NA, xlim = c(1,45), ylim = c(0, 11000), xlab = "Cycle",
     ylab = "Fluo")
 rect(Cq.range[1], 0, Cq.range[2], 11000, col = "cyan")
 for (i in 2L:ncol(reps384)) {
    lines(reps384[, 1], reps384[, i] - mean(reps384[1L:15, i]))
   }
dpcr_density(k = k.tmp, n = n.tmp)

## End(Not run)
```

`dpcr_density_gui` *Digital PCR Density Graphical User Interface*

Description

Launches graphical user interface that allows calculating density of positive partitions distribution.

Usage

```
dpcr_density_gui()
```

Warning

Any ad-blocking software may be cause of malfunctions.

Author(s)

Michal Burdukiewicz, Stefan Roediger.

See Also

dpcr_density.

`extract_dpcr` *Extract Digital PCR Experiment*

Description

Extract panel(s) or experiment(s) from a matrix while preserving all other attributes.

Usage

```
extract_dpcr(input, id)
```

Arguments

`input`	object of the class adpcr or ddpcr.
`id`	vector of indices or names of experiments or panels.

Details

The `extract_dpcr` function allows to choose one or more panels from an object of the adpcr or ddpcr class and save it without changing other attributes. It is the most recommended method of extracting a subset from an array of panels, because it preserves class and structure of the object in contrary to standard operator Extract.

Value

The object of the input's class (adpcr or ddpcr).

Note

The standard `Extract` operator `x[i]` treats dpcr objects as `matrix` and extracts values without preserving other attributies of the object.

Author(s)

Michal Burdukiewicz.

See Also

Opposite function: `bind_dpcr`

Examples

```
#sample extracting
panels <- sim_adpcr(10, 40, 1000, pos_sums = FALSE, n_panels = 50)
single_panel <- extract_dpcr(panels, 5)
random_three <- extract_dpcr(panels, sample.int(ncol(panels), 3))
all_but_one <- extract_dpcr(panels, -5)

#the same for fluorescence data
fluos <- sim_ddpcr(10, 40, 1000, pos_sums = FALSE, n_exp = 50, fluo = list(0.1, 0))
single_fluo <- extract_dpcr(fluos, 5)
```

---

limit_cq *Limit Cy0 values*

Description

The function `limit_cq` calculates the Cq values of a qPCR experiment within a defined range of cycles. The function can be used to extract Cq values of a chamber based qPCR for conversion into a dPCR experiment. All Cq values are obtained by Second Derivative Maximum or by Cy0 method (Guescini et al. (2008)).

Usage

```
limit_cq(data, cyc = 1, fluo = NULL, Cq_range = c(1, max(data[cyc])),
  model = l5, SDM = TRUE)
```

Arguments

`data`	a dataframe containing the qPCR data.
`cyc`	the column containing the cycle data. Defaults to first column.
`fluo`	the column(s) (runs) to be analyzed. If `NULL`, all runs will be considered (equivalent of `(1L:ncol(data))[-cyc]`).
`Cq_range`	is a user defined range of cycles to be used for the determination of the Cq values.
`model`	is the model to be used for the analysis for all runs. Defaults to 'l5' (see `pcrfit`).
`SDM`	if `TRUE`, Cq is approximated by the second derivative method. If `FALSE`, Cy0 method is used instead.

Details

The `Cq_range` for this function an be defined be the user. The default is to take all amplification curves into consideration. However, under certain circumstances it is recommended to define a range. For example if amplifications are positive in early cycle numbers (less than 10).

Approximated second derivative is influenced both by how often interpolation takes place in each data interval and by the smoothing method used. The user is encouraged to seek optimal parameters for his data himself. See `inder` for details.

The calculation of the Cy0 value (equivalent of Cq) is based on a five-parameter function. From experience this functions leads to good fitting and avoids overfitting of critical data sets. Regardless, the user is recommended to test for the optimal fitting function himself (see `mselect` for details).

Value

A data frame with two columns and number of rows equal to the number of runs analyzed. The column `Cy0` contains calculated Cy0 values. The column `in.range` contains adequate logical constant if given Cy0 value is in user-defined `Cq_range`.

Author(s)

Michal Burdukiewicz, Stefan Roediger.

References

Guescini M, Sisti D, Rocchi MB, Stocchi L & Stocchi V (2008) *A new real-time PCR method to overcome significant quantitative inaccuracy due to slight amplification inhibition*. BMC Bioinformatics, 9: 326.

Ruijter JM, Pfaffl MW, Zhao S, et al. (2013) *Evaluation of qPCR curve analysis methods for reliable biomarker discovery: bias, resolution, precision, and implications*. Methods, San Diego Calif 59:32–46.

See Also

SDM method: `inder`, `summary.der`.

Cy0 method: `mselect`, `efficiency`.

Examples

```
library(qpcR)
test <- cbind(reps[1L:45, ], reps2[1L:45, 2L:ncol(reps2)], reps3[1L:45,
      2L:ncol(reps3)])

# results.dPCR contains a column with the Cy0 values and a column with
# converted values.
Cq.range <- c(20, 30)
ranged <- limit_cq(data = test, cyc = 1, fluo = NULL,
                           Cq_range = Cq.range, model = l5)
# Same as above, but without Cq.range
no_range <- limit_cq(data = test, cyc = 1, fluo = NULL, model = l5)

# Same as above, but only three columns
no_range234 <- limit_cq(data = test, cyc = 1, fluo = c(2:4), model = l5)
```

many_peaks *Artificial data set with many peaks*

Description

A set of data from an artificial droplet flow experiment. It contains various kinds of peaks - positive peaks, negative peaks and peak-like noise.

Format

A data frame with 493 observations on the following 2 variables.

t a numeric vector

F fluorescence value

Examples

```
data(many_peaks)
plot(many_peaks, type = "b")
```

moments-methods *Calculate Moments of Poisson Distribution*

Description

The function allows user to quickly calculate moments of a Poisson distributions. The calculations are based on values of positive and total partitions or the theoretical lambda value.

Usage

```
moments(input)
```

Arguments

input	a single numeric object (lambda) or a two element vector (first element is treated as the number of positive partitions and the second as the number of total partitions) or an object of class adpcr or ddpcr.

Value

A named vector of four elements (mean, variance, skewness and kurtosis).

In case of adpcr or ddpcr object containing total number of positive molecules, a n-by-4 matrix, where n is the number of experiments.

In case of adpcr or ddpcr containing number of molecules per partition, a n*2-by-4 matrix, where n is the number of experiments. In this case empirical moments are calculated directly from a distribution of the data. Theoretical moments from a Poisson distribution with λ parameter taken from the data.

Note

Four first moments of a Poisson distribution.

Mean : λ.

Variance: λ.

Skewness: $\sqrt{\lambda}$.

Kurtosis: $\frac{1}{\lambda}$.

Author(s)

Michal Burdukiewicz.

Examples

```
# moments for lambda = 2
moments(2)

# moments for 100 positive partitions of 765 total partitions
moments(c(100, 765))

# calculate moments for an array digital PCR, total number of positive partitions
ddpcr1 <- sim_ddpcr(m = 10, n = 40, times = 50, pos_sums = TRUE, n_exp = 5)
moments(ddpcr1)

# calculate moments for an array digital PCR, detailed number of molecules in each partition
ddpcr2 <- sim_ddpcr(m = 10, n = 40, times = 50, pos_sums = FALSE, n_exp = 5)
moments(ddpcr2)
```

---

num2int *Convert numeric to integer*

Description

Converts `numeric` values to positive `integers` with a warning.

Usage

```
num2int(x)
```

Arguments

`x`	an `numeric` vector

Details

num2int uses `as.integer` functionality.

Examples

```
suppressWarnings(num2int(pi) == 3L)
```

pds *Plasmid dilution series results*

Description

These are the results data from the pds_raw data as calculated by the BioRad QX100 Droplet Digital PCR System.

Format

A data frame with 64 observations on the following 44 variables.

Well a factor with levels A01 to H04

ExptType a factor with levels Absolute Quantification

Experiment a factor with levels ABS

Sample a factor with levels B B + P 10^2 gDNA gDNA + P 10^0 gDNA + P 10^1 gDNA + P 10^-1 gDNA + P 10^2 gDNA + P 10^3 gDNA + P 10^4

TypeAssay a factor with levels Ch1NTC Ch1Unknown Ch2NTC Ch2Unknown

Assay a factor with levels ileS styA

Status a factor with levels Manual

Concentration a numeric vector

TotalConfMax a logical vector

TotalConfMin a logical vector

PoissonConfMax a numeric vector

PoissonConfMin a numeric vector

Positives a numeric vector

Negatives a numeric vector

Ch1.Ch2. a numeric vector

Ch1.Ch2..1 a numeric vector

Ch1.Ch2..2 a numeric vector

Ch1.Ch2..3 a numeric vector

Linkage a numeric vector

AcceptedDroplets a numeric vector

CNV a logical vector

TotalCNVMax a logical vector

TotalCNVMin a logical vector

PoissonCNVMax a logical vector

PoissonCNVMin a logical vector

ReferenceCopies a logical vector

UnknownCopies a logical vector

Ratio a numeric vector

TotalRatioMax a logical vector

TotalRatioMin a logical vector

PoissonRatioMax a numeric vector

PoissonRatioMin a numeric vector

FractionalAbundance a numeric vector

TotalFractionalAbundanceMax a logical vector

TotalFractionalAbundanceMin a logical vector

PoissonFractionalAbundanceMax a numeric vector

PoissonFractionalAbundanceMin a numeric vector

ReferenceAssayNumber a numeric vector

TargetAssayNumber a numeric vector

MeanAmplitudeofPositives a numeric vector

MeanAmplitudeofNegatives a numeric vector

MeanAmplitudeTotal a numeric vector

ExperimentComments a logical vector

MergedWells a logical vector

Details

Setup: Duplex assay with constant amount of genomic DNA and six 10-fold dilutions of plasmid DNA with 4 replicates, ranging theoretically from ~ 10^4 to 10^-1 copies/ micro L plus 4 replicates without plasmid DNA. Included are No-gDNA-control and No-template-control, 2 replicates each.

Annotation: FX.Y (X = dilution number, Y = replicate number). Hardware: Bio-Rad QX100 Droplet digital PCR system Details: Genomic DNA isolated from Pseudomonas putida KT2440. Plasmid is pCOM10-StyA::EGFP StyB [Jahn et al., 2013, Curr Opin Biotechnol, Vol. 24 (1): 79-87]. Template DNA was heat treated at 95 degree Celsius for 5 min prior to PCR. Channel 1, primers for genomic DNA marker ileS, Taqman probes (FAM labelled). Channel 2, primers for plasmid DNA marker styA, Taqman probes (HEX labelled).

Author(s)

Michael Jahn, Stefan Roediger, Michal Burdukiewcz

Source

Michael Jahn Flow cytometry group / Environmental microbiology Helmholtz Centre for Environmental Research - UFZ Permoserstrasse 15 / 04318 Leipzig / Germany phone +49 341 235 1318 michael.jahn [at] ufz.de / www.ufz.de

References

Jahn et al., 2013, *Curr Opin Biotechnol*, Vol. 24 (1): 79-87

Examples

```
## Not run:
# Loading libraries
library(lattice)
library(latticeExtra)
library(gplots)
library(Hmisc)
library(memisc)

dat <- pds
dat$Dilution <- rep(c(rep(c(10^(4:-1),0), each = 4),
       "NGC", "NGC", "NTC", "NTC"), 2)

print(xyplot(Concentration ~ factor(Dilution, levels=Dilution), dat,
      groups = Assay, ewidth = 0.08, ylim = c(10^-1.5, 10^4.5),
      ylab = "Concentration cp / micro L", xlab = "Theoretical plasmid
      concentration [cp/micro L]", strip = FALSE,
    scales = list(y = list(log = 10), alternating = FALSE),
    panel = function(x, y, groups = groups, ...) {
        panel.rect(7.5,-2, 10,5, col = grey(0.9), border = NA)
        panel.grid(h = -1,v = -1, col = grey(0.8), lty = 2)
        panel.xyplot(x, y, groups = groups, col = grey(0.6), ...)
        panel.key(c("IleS", "StyA"), corner = c(0.95,0.95))
        means <- tapply(y, list(x, groups), function(x) mean(x, na.rm = TRUE))
        stdev <- tapply(y, list(x, groups), function(x) sd(x, na.rm = TRUE))
        panel.errbars(1:9 + 0.25, y = cbind(means[, 1],
      means[, 1]-stdev[, 1], means[, 1] + stdev[, 1]),
      make.grid = "none", pch = 1,...)
        panel.errbars(1:9 + 0.25, y = cbind(means[, 2],
      means[, 2] - stdev[, 2], means[, 2] + stdev[, 2]),
      make.grid = "none", pch = 3, ...)
        panel.abline(a = c(5,-1), lty = 2, col = 2)
    }
  )
)

## End(Not run)
```

---

pds_raw *Plasmid dilution series raw data*

Description

These are the raw data from the pds_raw data set as measured by the BioRad QX100 Droplet Digital PCR System.

Format

The format is: List of 32 $ A01:'data.frame': 11964 obs. of 3 variables: ..$ Assay1.Amplitude: num [1:11964] 397 399 402 416 417 ... ..$ Assay2.Amplitude: num [1:11964] 3732 3808 4007

3778 3685 ... ..$ Cluster : int [1:11964] 4 4 4 4 4 4 4 4 4 4 ... $ A02:'data.frame': 11198 obs. of 3 variables: ..$ Assay1.Amplitude: num [1:11198] 310 429 433 445 445 ... ..$ Assay2.Amplitude: num [1:11198] 605 1092 994 1092 1140 ... ..$ Cluster : int [1:11198] 1 1 1 1 1 1 1 1 1 1 ... $ A03:'data.frame': 9672 obs. of 3 variables: ..$ Assay1.Amplitude: num [1:9672] 413 469 477 480 489 ... ..$ Assay2.Amplitude: num [1:9672] 781 1160 1205 1117 1098 ... ..$ Cluster : int [1:9672] 1 1 1 1 1 1 1 1 1 1 ... $ A04:'data.frame': 11901 obs. of 3 variables: ..$ Assay1.Amplitude: num [1:11901] 442 459 468 469 470 ... ..$ Assay2.Amplitude: num [1:11901] 3169 1161 1098 1064 1107 ... ..$ Cluster : int [1:11901] 4 1 1 1 1 1 1 1 1 1 ... $ B01:'data.frame': 11592 obs. of 3 variables: ..$ Assay1.Amplitude: num [1:11592] 438 449 451 453 454 ... ..$ Assay2.Amplitude: num [1:11592] 3996 4237 3910 3648 3832 ... ..$ Cluster : int [1:11592] 4 4 4 4 4 4 4 4 4 4 ... $ B02:'data.frame': 11715 obs. of 3 variables: ..$ Assay1.Amplitude: num [1:11715] 341 404 411 417 465 ... ..$ Assay2.Amplitude: num [1:11715] 705 1013 892 936 996 ... ..$ Cluster : int [1:11715] 1 1 1 1 1 1 1 1 1 1 ... $ B03:'data.frame': 11194 obs. of 3 variables: ..$ Assay1.Amplitude: num [1:11194] 31.2 266.9 281.6 286.1 300.3 ... ..$ Assay2.Amplitude: num [1:11194] 668 555 508 585 571 ... ..$ Cluster : int [1:11194] 1 1 1 1 1 1 1 1 1 1 ... $ B04:'data.frame': 12813 obs. of 3 variables: ..$ Assay1.Amplitude: num [1:12813] 380 464 474 483 487 ... ..$ Assay2.Amplitude: num [1:12813] 830 913 1143 1157 1032 ... ..$ Cluster : int [1:12813] 1 1 1 1 1 1 1 1 1 1 ... $ C01:'data.frame': 10903 obs. of 3 variables: ..$ Assay1.Amplitude: num [1:10903] 427 442 443 444 445 ... ..$ Assay2.Amplitude: num [1:10903] 3803 3832 3634 3899 3932 ... ..$ Cluster : int [1:10903] 4 4 4 4 4 4 4 4 4 4 ... $ C02:'data.frame': 9638 obs. of 3 variables: ..$ Assay1.Amplitude: num [1:9638] 442 446 454 454 457 ... ..$ Assay2.Amplitude: num [1:9638] 1107 1131 3644 881 3460 ... ..$ Cluster : int [1:9638] 1 1 4 1 4 4 1 1 1 1 ... $ C03:'data.frame': 12194 obs. of 3 variables: ..$ Assay1.Amplitude: num [1:12194] 461 466 470 475 475 ... ..$ Assay2.Amplitude: num [1:12194] 842 1089 1156 1115 1194 ... ..$ Cluster : int [1:12194] 1 1 1 1 1 1 1 1 1 1 ... $ C04:'data.frame': 10889 obs. of 3 variables: ..$ Assay1.Amplitude: num [1:10889] 343 448 456 466 474 ... ..$ Assay2.Amplitude: num [1:10889] 633 1149 1073 1161 1089 ... ..$ Cluster : int [1:10889] 1 1 1 1 1 1 1 1 1 1 ... $ D01:'data.frame': 11196 obs. of 3 variables: ..$ Assay1.Amplitude: num [1:11196] 110 413 417 435 444 ... ..$ Assay2.Amplitude: num [1:11196] 734 3752 3736 3885 3720 ... ..$ Cluster : int [1:11196] 1 4 4 4 4 4 4 4 4 4 ... $ D02:'data.frame': 12013 obs. of 3 variables: ..$ Assay1.Amplitude: num [1:12013] 453 457 459 463 462 ... ..$ Assay2.Amplitude: num [1:12013] 1113 1178 1054 1088 3108 ... ..$ Cluster : int [1:12013] 1 1 1 1 4 1 4 1 1 4 ... $ D03:'data.frame': 11126 obs. of 3 variables: ..$ Assay1.Amplitude: num [1:11126] 323 460 463 471 473 ... ..$ Assay2.Amplitude: num [1:11126] 661 1138 1103 1091 1138 ... ..$ Cluster : int [1:11126] 1 1 1 1 1 1 1 1 1 1 ... $ D04:'data.frame': 12793 obs. of 3 variables: ..$ Assay1.Amplitude: num [1:12793] 363 433 442 459 460 ... ..$ Assay2.Amplitude: num [1:12793] 703 1065 1071 1060 1119 ... ..$ Cluster : int [1:12793] 1 1 1 1 1 1 1 1 1 1 ... $ E01:'data.frame': 11823 obs. of 3 variables: ..$ Assay1.Amplitude: num [1:11823] 368 434 448 453 462 ... ..$ Assay2.Amplitude: num [1:11823] 778 3751 3585 1125 3797 ... ..$ Cluster : int [1:11823] 1 4 4 1 4 4 4 1 1 4 ... $ E02:'data.frame': 12046 obs. of 3 variables: ..$ Assay1.Amplitude: num [1:12046] 268 413 414 454 455 ... ..$ Assay2.Amplitude: num [1:12046] 582 3738 2412 1071 1076 ... ..$ Cluster : int [1:12046] 1 4 4 1 1 1 1 1 1 1 ... $ E03:'data.frame': 11026 obs. of 3 variables: ..$ Assay1.Amplitude: num [1:11026] 357 446 456 456 460 ... ..$ Assay2.Amplitude: num [1:11026] 675 1138 1095 1145 1138 ... ..$ Cluster : int [1:11026] 1 1 1 1 1 1 1 1 1 1 ... $ E04:'data.frame': 12838 obs. of 3 variables: ..$ Assay1.Amplitude: num [1:12838] 460 467 472 477 482 ... ..$ Assay2.Amplitude: num [1:12838] 1207 1238 1143 3754 1153 ... ..$ Cluster : int [1:12838] 1 1 1 4 1 1 1 4 4 1 ... $ F01:'data.frame': 12173 obs. of 3 variables: ..$ Assay1.Amplitude: num [1:12173] 354 389 457 456 466 ... ..$ Assay2.Amplitude: num [1:12173] 739 2714 3888 3775 3857 ... ..$ Cluster : int [1:12173] 1 4 4 4 4 4 4 1 1 4 ... $ F02:'data.frame':

13786 obs. of 3 variables: ..$ Assay1.Amplitude: num [1:13786] 338 359 439 459 461 ... ..$ Assay2.Amplitude: num [1:13786] 525 638 674 3891 1046 ... ..$ Cluster : int [1:13786] 1 1 1 4 1 1 1 1 1 1 ... $ F03:'data.frame': 11249 obs. of 3 variables: ..$ Assay1.Amplitude: num [1:11249] 431 445 447 448 455 ... ..$ Assay2.Amplitude: num [1:11249] 1090 3330 1048 1187 1098 ... ..$ Cluster : int [1:11249] 1 4 1 1 1 1 1 1 1 1 ... $ F04:'data.frame': 12076 obs. of 3 variables: ..$ Assay1.Amplitude: num [1:12076] 407 424 436 446 447 ... ..$ Assay2.Amplitude: num [1:12076] 699 1047 3683 1085 1088 ... ..$ Cluster : int [1:12076] 1 1 4 1 1 1 1 1 1 1 ... $ G01:'data.frame': 10188 obs. of 3 variables: ..$ Assay1.Amplitude: num [1:10188] 460 468 470 471 482 ... ..$ Assay2.Amplitude: num [1:10188] 3813 3213 3949 3658 2202 ... ..$ Cluster : int [1:10188] 4 4 4 4 4 4 4 1 4 4 ... $ G02:'data.frame': 11018 obs. of 3 variables: ..$ Assay1.Amplitude: num [1:11018] 445 452 460 460 461 ... ..$ Assay2.Amplitude: num [1:11018] 1064 1090 1054 1087 1116 ... ..$ Cluster : int [1:11018] 1 1 1 1 1 1 1 1 1 1 ... $ G03:'data.frame': 12073 obs. of 3 variables: ..$ Assay1.Amplitude: num [1:12073] 294 459 460 468 489 ... ..$ Assay2.Amplitude: num [1:12073] 658 1105 1168 1160 1115 ... ..$ Cluster : int [1:12073] 1 1 1 1 1 1 1 1 1 1 ... $ G04:'data.frame': 12320 obs. of 3 variables: ..$ Assay1.Amplitude: num [1:12320] 444 476 484 491 498 ... ..$ Assay2.Amplitude: num [1:12320] 831 1158 984 1237 1325 ... ..$ Cluster : int [1:12320] 1 1 1 1 1 1 1 1 1 1 ... $ H01:'data.frame': 12271 obs. of 3 variables: ..$ Assay1.Amplitude: num [1:12271] 377 388 404 409 433 ... ..$ Assay2.Amplitude: num [1:12271] 2884 917 3679 830 2780 ... ..$ Cluster : int [1:12271] 4 1 4 1 4 4 4 1 4 4 ... $ H02:'data.frame': 12595 obs. of 3 variables: ..$ Assay1.Amplitude: num [1:12595] 364 411 419 421 424 ... ..$ Assay2.Amplitude: num [1:12595] 691 836 2885 812 1177 ... ..$ Cluster : int [1:12595] 1 1 4 1 1 1 1 1 1 1 ... $ H03:'data.frame': 13905 obs. of 3 variables: ..$ Assay1.Amplitude: num [1:13905] 314 400 426 431 433 ... ..$ Assay2.Amplitude: num [1:13905] 509 987 1037 1074 1053 ... ..$ Cluster : int [1:13905] 1 1 1 1 1 1 1 1 1 1 ... $ H04:'data.frame': 12972 obs. of 3 variables: ..$ Assay1.Amplitude: num [1:12972] 344 446 461 467 482 ... ..$ Assay2.Amplitude: num [1:12972] 641 869 987 789 1117 ... ..$ Cluster : int [1:12972] 1 1 1 1 1 1 1 1 1 1 ...

Details

The results can be as calculated by the BioRad QX100 Droplet Digital PCR System are to be found in pds.

Setup: Duplex assay with constant amount of genomic DNA and six 10-fold dilutions of plasmid DNA with 4 replicates, ranging theoretically from ~ 10^4 to 10^-1 copies/ micro L plus 4 replicates without plasmid DNA. Included are No-gDNA-control and No-template-control, 2 replicates each.

Annotation: FX.Y (X = dilution number, Y = replicate number). Hardware: Bio-Rad QX100 Droplet digital PCR system Details: Genomic DNA isolated from Pseudomonas putida KT2440. Plasmid is pCOM10-StyA::EGFP StyB [Jahn et al., 2013, Curr Opin Biotechnol, Vol. 24 (1): 79-87]. Template DNA was heat treated at 95 degree Celsius for 5 min prior to PCR. Channel 1, primers for genomic DNA marker ileS, Taqman probes (FAM labelled). Channel 2, primers for plasmid DNA marker styA, Taqman probes (HEX labelled).

Author(s)

Michael Jahn, Stefan Roediger, Michal Burdukiewcz

Source

Michael Jahn Flow cytometry group / Environmental microbiology Helmholtz Centre for Environmental Research - UFZ Permoserstrasse 15 / 04318 Leipzig / Germany phone +49 341 235 1318

michael.jahn [at] ufz.de / www.ufz.de

References

Jahn et al., 2013, *Curr Opin Biotechnol*, Vol. 24 (1): 79-87

Examples

```
#str(pds_raw)
bioamp(data = pds_raw[["A01"]], main = "Well A01", pch = 19)
```

`plot.qpcrpp` *Plot* `qpcrpp` *objects*

Description

An analytical plot describing relationship between the cycle number and the current value of Poisson mean. The plot can be used for quality control of process.

Arguments

`x`	is a `qpcrpp` object.
`mincyc`	is the first cycle to start the plot from.
`maxcyc`	the last cycle for the plot.
`rug`	Adds a rug representation of the data to the plot.
`digits`	how many significant digits are to be used in the plot.

Details

The `rug` parameter allows user to add density of the number of events to the plot.

Author(s)

Stefan Roediger, Michal Burdukiewicz

See Also

`qpcrpp`

Examples

```
library(qpcR)
test <- cbind(reps[1L:45, ], reps2[1L:45, 2L:ncol(reps2)], reps3[1L:45,
        2L:ncol(reps3)])
ranged <- limit_cq(data = test, cyc = 1, fluo = NULL, model = l5)

plot(qpcr2pp(ranged[, 1], ranged[, 2], delta = 5), rug = TRUE)
```

plot_panel *Plot Panel*

Description

The plot_panel function takes objects of the class adpcr to enable customizable graphical representations of a chamber-based digital PCR experiments (e.g., Digital Array (R) IFCs (integrated fluidic circuits) of the BioMark (R) and EP1 (R)).

Usage

```
plot_panel(input, nx_a, ny_a, col = "red", legend = TRUE, half = "none",
  use_breaks = TRUE, ...)
```

Arguments

input	object of the adpcr class. See Details.
nx_a	Number of columns in a plate.
ny_a	Number of rows in a plate.
col	A single color or vector of colors for each level of input.
legend	If TRUE, a built-in legend is added to the plot.
half	If left or right, every well is represented only by the adequate half of the rectangle.
use_breaks	if TRUE, input is cut into intervals using breaks slot. If FALSE, input is converted to factor using as.factor.
...	Arguments to be passed to plot function.

Details

Currently only objects containing just one column of data (one panel) can be plotted (see Examples how easily plot multipanel objects). Moreover, the object must contain fluorescence intensities or exact number of molecules or the positive hits derived from the Cq values for each well. The Cq values can be obtained by custom made functions (see example in dpcr_density)) or the yet to implement "qpcr_analyser function from the dpcR package.

If the col argument has length one, a color is assigned for each interval of the input, with the brightest colors for the lowest values.

Value

A list of coordinates of each microfluidic well and an assigned color.

Author(s)

Michal Burdukiewicz, Stefan Roediger.

See Also

extract_dpcr.

Examples

```
# Create a sample dPCR experiment with 765 elements (~> virtual compartments)
# of target molecule copies per compartment as integer numbers (0,1,2)
ttest <- sim_adpcr(m = 400, n = 765, times = 20, pos_sums = FALSE,
                   n_panels = 1)
# Plot the dPCR experiment results with default settings
plot_panel(ttest, nx_a = 45, ny_a = 17)

#do it without breaks
plot_panel(ttest, nx_a = 45, ny_a = 17, use_breaks = FALSE)

# Apply a binary color code with blue as positive
slot(ttest, "breaks") <- c(0, 2, 4)
plot_panel(ttest, nx_a = 45, ny_a = 17, col = "blue")

# Apply a two color code for number of copies per compartment
plot_panel(ttest, nx_a = 45, ny_a = 17, col = c("blue", "red"))



# supply customized breaks and compare
par(mfcol = c(2, 1))
plot_panel(ttest, nx_a = 45, ny_a = 17)
slot(ttest, "breaks") <- c(0, 1, 2, (max(slot(ttest, "breaks")) + 1))
plot_panel(ttest, nx_a = 45, ny_a = 17)
par(mfcol = c(1, 1))

# plot few panels
ttest2 <- sim_adpcr(m = 400, n = 765, times = 40, pos_sums = FALSE,
                    n_panels = 4)
par(mfcol = c(2, 2))
four_panels <- lapply(1:ncol(ttest2), function(i)
       plot_panel(extract_dpcr(ttest2, i), nx_a = 45, ny_a = 17, legend = FALSE,
         main = paste("Panel", LETTERS[i], sep = " ")))
par(mfcol = c(1, 1))

# two different channels
plot_panel(extract_dpcr(ttest2, 1), nx_a = 45, ny_a = 17, legend = FALSE,
           half = "left")
par(new = TRUE)
plot_panel(extract_dpcr(ttest2, 2), nx_a = 45, ny_a = 17, col = "blue",
           legend = FALSE, half = "right")

# plot two panels with every well as only the half of the rectangle
ttest3 <- sim_adpcr(m = 400, n = 765, times = 40, pos_sums = FALSE,
                    n_panels = 2)
par(mfcol = c(1, 2))
two_panels <- lapply(1:ncol(ttest3), function(i)
```

```
        plot_panel(extract_dpcr(ttest3, i), nx_a = 45, ny_a = 17, legend = FALSE,
          main = paste("Panel", LETTERS[i], sep = " ")))
par(mfcol = c(1, 1))
```

`plot_vic_fam` *Amplitude Plot VIC and FAM Channels of a Droplet Digital PCR Experiment*

Description

This function generates an amplitude plot of two fluorescence channels as found in droplet digital PCR.

Usage

```
plot_vic_fam(vic, fam, col_vic = "green", col_fam = "blue", circle = TRUE)
```

Arguments

`vic`	Amplitudes of the VIC channel - object of class `ddpcr`.
`fam`	Amplitudes of the FAM channel - object of class `ddpcr`.
`col_vic`	Color of the VIC channel.
`col_fam`	Color of the FAM channel.
`circle`	If TRUE circles are drawn, if FALSE not. If "numeric", specifies the radius of circles.

Details

Droplet digital PCR experiments consist of three steps (droplet generation, clonal amplification, droplet amplitude analysis). Typically 20000 nano-sized droplets are analyzed and separated into amplification-positive and amplification-negative droplets. An example of such system is the Bio-Rad QX100 and QX200 (Pinheiro et al. 2012). Such systems have applications in the detection of rare DNA target copies, the determination of copy number variations (CNV), detection of mutation, or expression analysis of genes or miRNA. Each droplet is analyzed individually using a virtual two-color detection system. The channels are treated separately but finally aligned (e.g., FAM and VIC or FAM and HEX).

Author(s)

Michal Burdukiewicz, Stefan Roediger.

References

Pinheiro, L.B., Coleman, V.A., Hindson, C.M., Herrmann, J., Hindson, B.J., Bhat, S., and Emslie, K.R. (2012). *Evaluation of a droplet digital polymerase chain reaction format for DNA copy number quantification*. Anal. Chem. 84, 1003 - 1011.

Examples

```
# Generate an amplitude plot for the first fluorescence channel (e.g., FAM)
fluos1 <- sim_ddpcr(m = 16, n = 30, times = 100, pos_sums = FALSE, n_exp = 1,
  fluo = list(0.1, 0))

# Generate an amplitude plot for the second fluorescence channel (e.g., VIC)
fluos2 <- sim_ddpcr(m = 16, n = 30, times = 100, pos_sums = FALSE, n_exp = 1,
  fluo = list(0.1, 0))

# Plot the amplitudes of both fluorescence channel in an aligned fashion
plot_vic_fam(fam = fluos1, vic = fluos2)

# Same as above but different colors
plot_vic_fam(fam = fluos1, vic = fluos2, col_vic = "red", col_fam = "yellow")

# Same as above without circles
plot_vic_fam(fam = fluos1, vic = fluos2, col_vic = "red", col_fam = "yellow", circle = FALSE)

# Generate two channels in one object and plot them
fluos_both <- sim_ddpcr(m = 16, n = 30, times = 100, pos_sums = FALSE, n_exp = 2,
  fluo = list(0.1, 0))
plot_vic_fam(extract_dpcr(fluos_both, 1), extract_dpcr(fluos_both, 2))
```

---

qpcr2pp *qPCR to Poisson Process*

Description

Selected platforms (e.g., Open Array) are real-time platforms. dPCR can be described by Poisson statistics. The function qpcr2pp takes a step further and interprets the dPCR as a Poisson process if it is analyzed as a "time" based process.

Usage

```
qpcr2pp(cycles, process, data = NULL, NuEvents = 1, delta = 1)
```

Arguments

cycles	the column containing the cycle data. Defaults to first column.
process	the column containing fluorescence values.
data	a dataframe containing the qPCR data.
NuEvents	NuEvents is "number of expected events" within a time frame (interval).
delta	is the difference "time (cycles) points" e.g., Cycle 18 and 25.

Details

The dPCR Technology breaks fundamentally with the previous concept of nucleic acid quantification. dPCR can be seen as a next generation nucleic acid quantification method based on PCR. The key difference between dPCR and traditional PCR lies in the method of measuring (absolute) nucleic acids amounts. This is possible after “clonal DNA amplification” in thousands of small separated partitions (e.g., droplets, nano chambers). Partitions with no nucleic acid remain negative and the others turn positive. Selected technologies (e.g., OpenArray(R) Real-Time PCR System) monitor amplification reactions in the chambers in real-time. Cq values are calculated from the amplification curves and converted into discrete events by means of positive and negative partitions and the absolute quantification of nucleic acids is done by Poisson statistics.

PCR data derived from a qPCR experiment can be seen as a series of events over time. We define t_i as the time between the first (i - 1)^st and the i^th event. Therefore, the time S_n is the sum of all t_i from $i = 1$ to $i = n$. This is the time to the n^th event. $S(t)$ is the number of events in $[0, t]$. This can be seen as a Poisson process. The Poisson statistics is the central theorem to random processes in digital PCR.

The function `qpcr2pp` is used to model random point events in time units (PCR cycles), such as the increase of signal during a qPCR reaction in a single compartment. A Poisson process can be used to model times at which an event occurs in a "system". The `qpcr2pp` (quantitative Real-Time PCR to Poisson process) function transforms the qPCR amplification curve data to quantification points (Cq) which are visualized as Poisson process. This functions helps to spot differences between replicate runs of digital PCR experiments. In ideal scenarios the `qpcr2pp` plots are highly similar.

This tool might help to spot differences between experiments (e.g., inhibition of amplification reactions, influence of the chip arrays). The qPCR is unique because the amplification of conventional qPCRs takes place in discrete steps (cycles: 1, 2 ... 45), but the specific Cq values are calculated with continuous outcomes (Cq: 18.2, 25.7, ...). Other amplification methods such as isothermal amplifications are time based and thus better suited for Poisson process.

Value

An object of `qpcrpp` class.

Author(s)

Stefan Roediger, Michal Burdukiewicz.

Examples

```
library(qpcR)
test <- cbind(reps[1L:45, ], reps2[1L:45, 2L:ncol(reps2)],
      reps3[1L:45, 2L:ncol(reps3)])

# before interpolation qPCR experiment must be converted into dPCR
Cq.range <- c(20, 30)
ranged <- limit_cq(data = test, cyc = 1, fluo = NULL,
                     Cq_range = Cq.range, model = l5)

qpcr2pp(ranged[,1], ranged[,2], delta = 5)
```

qpcrpp-class *Class* "qpcrpp"

Description

An object representing digital PCR reaction depicted as Poisson process.

Slots

list(".Data") "matrix" with three columns containing: number of cycles, amplification curves and cumulative sum of events.

: "matrix" with three columns containing: number of cycles, amplification curves and cumulative sum of events.

list("mu") "numeric" of the expected number of events in defined interval.

: "numeric" of the expected number of events in defined interval.

list("CT") "numeric" value of the "average time" between the occurence of a positive reaction and another positive reaction.

: "numeric" value of the "average time" between the occurence of a positive reaction and another positive reaction.

list("partitions") "integer" value equal to the number of partitions.

: "integer" value equal to the number of partitions.

list("events") "integer" value equal number of events (positive partitions taken to further analysis)

: "integer" value equal number of events (positive partitions taken to further analysis)

Author(s)

Stefan Roediger, Michal Burdukiewicz.

See Also

plot.qpcrpp,

qpcr_analyser *qPCR Analyser*

Description

Calculate statistics based on fluorescence. The function can be used to analyze amplification curve data from quantitative real-time PCR experiments. The analysis includes the fitting of the amplification curve by a non-linear function and the calculation of a quantification point (often referred to as Cp (crossing-point), Cq or Ct) based on a user defined method. The function can be used to analyze data from chamber based dPCR machines.

Arguments

input	a dataframe containing the qPCR data or a result of function modlist or an object of the class adpcr.
cyc	the column containing the cycle data. Defaults to first column.
fluo	the column(s) (runs) to be analyzed. If NULL, all runs will be considered. Use fluo = 2 to chose the second column for example.
model	is the model to be used for the analysis for all runs. Defaults to '15' (see pcrfit).
norm	logical. Indicates if the raw data should be normalized within [0, 1] before model fitting.
iter_tr	iter_tr number of iteration to fit the curve.
type	is the method for the crossing point/threshold cycle estimation and efficiency estimation (efficiency). Defaults to 'Cy0' (Cy0).
takeoff	logical; if TRUE calculates the first significant cycle of the exponential region (takeoff point). See takeoff for details.

Details

The qpcr_analyzer is a functions to automatize the analysis of amplification curves from conventional quantitative real-time PCR (qPCR) experiments and is adapted for the needs in dPCR. This function calls instances of the the qpcR package to calculate the quantification points (cpD1, cpD2, Cy0 (default), TOP (optional)), the amplification efficiency, fluorescence at the quantification point (Cq), the absolute change of fluorescence and the take-off point (TOP). Most of the central functionality of the qpcR package is accessible. The user can assign concentrations to the samples. One column contains binary converted (pos (1) and neg (0)) results for the amplification reaction based on a user defined criteria (Cq-range, fluorescence cut-off, ...). qpcr_analyser tries to detect cases where an amplification did not take place of was impossible to analyze. By default qpcr_analyser analyses uses the Cy0 as described in Guescini et al. (2008) for estimation of the quantification point since method is considered to be better suited for many probe systems. By default a 5-parameter model is used to fit the amplification curves. As such qpcr_analyser is a functions which serves for preliminary data inspection (see Example section) and as input for other R functions from the dpcR package (e.g., plot_panel).

Value

A matrix where each column represents crossing point, efficiency, the raw fluorescence value at the point defined by type and difference between minimum and maximum of observed fluorescence. If takeoff parameter is TRUE, additional two column represents start and the end of the fluorescence growth.

Author(s)

Stefan Roediger, Andrej-Nikolai Spiess, Michal Burdukiewicz.

References

Ritz C, Spiess An-N, *qpcR: an R package for sigmoidal model selection in quantitative real-time polymerase chain reaction analysis*. Bioinformatics 24 (13), 2008.

Andrej-Nikolai Spiess (2013). qpcR: Modelling and analysis of real-time PCR data. http://CRAN.R-project.org/package=qpcR

See Also

modlist.

Examples

```
# Take data of guescini1 data set from the qpcR R package.
library(qpcR)
# Use the first column containing the cycles and the second column for sample F1.1.
data(guescini1)
qpcr_analyser(guescini1, cyc = 1, fluo = 2)

# Use similar setting as before but set takeoff to true for an estimation of
# the first significant cycle of the exponential region.
qpcr_analyser(guescini1, cyc = 1, fluo = 2, takeoff = TRUE)

# Use similar setting as before but use qpcr_analyser in a loop to calculate the results for the
# first four columns containing the fluorescence in guescini1
print(qpcr_analyser(guescini1, cyc = 1, fluo = 2:5, takeoff = TRUE))

# Run qpcr_analyser on the list of models (finer control on fitting model process)
models <- modlist(guescini1)
qpcr_analyser(models)
```

---

rtadpcr-class *Class* "rtadpcr" - *real-time array digital PCR experiments*

Description

A class designed to contain results from real-time array digital PCR experiments. Data is represented as matrix, where each column describes different measurement point (i.e. cycle number) and every row different partition.

Slots

list(".Data") "matrix" containing data from array. See Description.

: "matrix" containing data from array. See Description.

list("n") Object of class "integer" equal to the number of partitions.

: Object of class "integer" equal to the number of partitions.

list("type") Object of class "character" defining type of data.

: Object of class "character" defining type of data.

Author(s)

Michal Burdukiewicz.

See Also

End-point array digital PCR: adpcr.

Droplet digital PCR: ddpcr.

Examples

```
#none
```

---

show-methods *Methods for Function* show

Description

Expands function show allowing showing objects of the class adpcr to or ddpcr.

Arguments

object	an object of class adpcr or ddpcr.

Author(s)

Michal Burdukiewicz.

Examples

```
#array dpcr
ptest <- sim_adpcr(400, 765, 5, FALSE, n_panels = 1)
show(ptest)

#multiple experiments
ptest <- sim_adpcr(400, 765, 5, FALSE, n_panels = 5)
show(ptest)

#droplet dpcr - fluorescence
dropletf <- sim_ddpcr(7, 20, times = 5, fluo = list(0.1, 0))
show(dropletf)

#droplet dpcr - number of molecules
droplet <- sim_ddpcr(7, 20, times = 5)
show(droplet)
```

sim_adpcr *Simulate Array Digital PCR*

Description

A function that simulates results of an array digital PCR.

Usage

```
sim_adpcr(m, n, times, n_panels = 1, dube = FALSE, pos_sums = FALSE)
```

Arguments

m	the total number of template molecules added to the plate. Must be a positive integer.
n	the number of chambers per plate. Must be a positive integer.
times	number of repetitions (see Details).
n_panels	the number of panels that are simulated by the function. Cannot have higher value than the times argument.
dube	if TRUE, the function is strict implementation of array digital PCR simulation (as in Dube et al., 2008). If FALSE, the function calculates only approximation of Dube's experiment. See Details and References.
pos_sums	if TRUE, function returns only the total number of positive (containing at least one molecule) chamber per panel. If FALSE, the functions returns a vector of length equal to the number of chambers. Each element of the vector represents the number of template molecules in a given chamber.

Details

The array digital PCR is performed on plates containing many microfluidic chambers with a randomly distributed DNA template, fluorescence labels and standard PCR reagents. After the amplification reaction, performed independently in each chamber, the chambers with the fluorescence level below certain threshold are treated as negative. From differences between amplification curves of positive chambers it is possible to calculate both total number of template molecules as well as their approximate number in a single chamber.

The function contains two implementations of the array digital PCR simulation. First one was described in Dube at. al (2008). This method is based on random distributing $m \times times$ molecules between $n \times times$ chambers. After this step, the required number of plates is created by the random sampling of chambers without replacement. The above method is used, when the dube argument has value TRUE.

The second method treats the total number of template molecules as random variable with a normal distribution $\mathcal{N}(n, 0.05n)$. The exact sum of total molecules per plate is calculated and randomly adjusted to the value of $m \times times$. The above method is used, when the dube argument has value FALSE. This implementation is much faster than previous one, especially for big simulations. The higher the value of the argument times, the simulation result is closer to theoretical calculations.

Value

If the pos_sums argument has value FALSE, the function returns a matrix with n rows and n_panels columns. Each column represents one plate. Type of such simulation would be "nm". If the pos_sums argument has value TRUE, the function returns a matrix with one row and n_panels columns. Each column contains the total number of positive chambers in each plate and type of simulation would be set as "tp".

In each case the value is an object of the adpcr class.

Author(s)

Michal Burdukiewicz.

References

Dube S, Qin J, Ramakrishnan R, *Mathematical Analysis of Copy Number Variation in a DNA Sample Using Digital PCR on a Nanofluidic Device*. PLoS ONE 3(8), 2008.

See Also

sim_ddpcr.

Examples

```
# Simulation of a digital PCR experiment with a chamber based technology.
# The parameter pos_sums was altered to change how the total number of positive
# chamber per panel are returned. An alteration of the parameter has an impact
# in the system performance.
adpcr_big <- sim_adpcr(m = 10, n = 40, times = 1000, pos_sums = FALSE, n_panels = 1000)
adpcr_small <- sim_adpcr(m = 10, n = 40, times = 1000, pos_sums = TRUE, n_panels = 1000)
# with pos_sums = TRUE, output allocates less memory
object.size(adpcr_big)
object.size(adpcr_small)

# Mini version of Dube et al. 2008 experiment, full requires replic <- 70000
# The number of replicates was reduced by a factor of 100 to lower the computation time.
replic <- 700
dube <- sim_adpcr(400, 765, times = replic, dube = TRUE,
    pos_sums = TRUE, n_panels = replic)
mean(dube) # 311.5616
sd(dube) # 13.64159

# Create a barplot from the simulated data similar to Dube et al. 2008
bp <- barplot(table(factor(dube, levels = min(dube):max(dube))),
      space = 0)
lines(bp, dnorm(min(dube):max(dube), mean = 311.5, sd = 13.59)*replic,
      col = "green", lwd = 3)

# Exact Dube's method is a bit slower than other one, but more accurate
system.time(dub <- sim_adpcr(m = 400, n = 765, times = 500, n_panels = 500,
  pos_sums = TRUE))
system.time(mul <- sim_adpcr(m = 400, n = 765, times = 500, n_panels = 500,
```

```
  pos_sums = FALSE))
```

`sim_ddpcr` *Simulate Droplet Digital PCR*

Description

A function that simulates results of a droplet digital PCR.

Usage

```
sim_ddpcr(m, n, times, n_exp = 1, dube = FALSE, pos_sums = FALSE,
  fluo = NULL)
```

Arguments

`m`	the total number of template molecules added to the plate. Must be a positive integer.
`n`	the number of chambers per plate. Must be a positive integer.
`times`	number of repetitions (see Details).
`n_exp`	the number of experiments that are simulated by the function. Cannot have higher value than the `times` argument.
`dube`	if `TRUE`, the function is strict implementation of digital PCR simulation (as in Dube et al., 2008). If `FALSE`, the function calculates only approximation of Dube's experiment. See Details and References.
`pos_sums`	if `TRUE`, function returns only the total number of positive (containing at least one molecule) chamber per panel. If `FALSE`, the functions returns a vector of length equal to the number of chambers. Each element of the vector represents the number of template molecules in a given chamber.
`fluo`	if `NULL`, the function calculates number of molecules per well or total number of positive droplets. If list of two, the first argument defines smoothness of the fluorescence curve and second space between two consecutive measured droplets. Space must be a vector containing positive integers of the length `n` or 1.

Details

implementations of the array digital PCR simulation. First one was described in Dube at. al (2008). This method is based on random distributing $m \times times$ molecules between $n \times times$ chambers. After this step, the required number of plates is created by the random sampling of chambers without replacement. The above method is used, when the `dube` argument has value `TRUE`.

The higher the value of the argument `times`, the simulation result is closer to theoretical calculations.

Value

If the pos_sums argument has value FALSE, the function returns matrix with n rows and n_panels columns. Each column represents one plate. Type of such simulation would be "nm". If the pos_sums argument has value TRUE, the function return matrix with one row and n_panels columns. Each column contains the total number of positive chambers in each plate and type of simulation would be set as "tp".

In each case the value is an object of the ddpcr class.

Note

Although Dube's simulation of digital PCR was developed for array digital PCR, it's also viable for simulating droplet-based methods.

Author(s)

Michal Burdukiewicz, Stefan Roediger.

See Also

sim_adpcr.

Examples

```
#simulate fluorescence data
tmp_VIC <- sim_ddpcr(m = 7, n = 20, times = 5, fluo = list(0.1, 0))
tmp_FAM <- sim_ddpcr(m = 15, n = 20, times = 5, fluo = list(0.1, 0))
par(mfrow = c(2,1))
plot(tmp_VIC, col = "green", type = "l")
plot(tmp_FAM, col = "blue", type = "l")
summary(tmp_FAM)

summary(sim_ddpcr(m = 7, n = 20, times = 5, n_exp = 5))
```

---

summary-methods *Methods for Function* summary

Description

Expands function summary allowing printing summaries objects of the class adpcr to or ddpcr.

Arguments

object	an object of class adpcr or ddpcr.
print	if FALSE, no output is printed, only calculations are performed.

Details

The function prints summary of the dPCR reaction, including k (number of positive chambers), n (total number of chambers), estimated lambda and m (number of molecules per plate), as well as confidence intervals for the last two variables.

Value

The data frame with estimated values of lambda, m and corresponding confidence intervals.

Note

If summary is used on an object containing results of many experiments, all experiments would be independently summarized. Currently supported only for objects of class adpcr.

Author(s)

Michal Burdukiewicz, Stefan Roediger.

References

Bhat S, Herrmann J, Corbisier P, Emslie K, *Single molecule detection in nanofluidic digital array enables accurate measurement of DNA copy number*. Analytical and Bioanalytical Chemistry 2 (394), 2009.

Dube S, Qin J, Ramakrishnan R, *Mathematical Analysis of Copy Number Variation in a DNA Sample Using Digital PCR on a Nanofluidic Device*. PLoS ONE 3(8), 2008.

Examples

```
# array dpcr
# Simulates a chamber based digital PCR with m total number of template molecules
# and n number of chambers per plate and assigns it as object ptest of the class
# adpcr for a single panel. The summary function on ptest gets assigned to summ
# and the result with statistics according to Dube et al. 2008 and Bhat et al. 2009
# gets printed.
ptest <- sim_adpcr(m = 400, n = 765, times = 5, dube = FALSE, n_panels = 1)
summ <- summary(ptest) #save summary
print(summ)

# multiple experiments
# Similar to the previous example but with five panels
ptest <- sim_adpcr(m = 400, n = 765, times = 5, dube = FALSE, n_panels = 5)
summary(ptest)

# droplet dpcr - fluorescence
# Simulates a droplet digital PCR with m = 7 total number of template molecules
# and n = 20 number of  droplets. The summary function on dropletf gives the
# statistics according to Dube et al. 2008 and Bhat et al. 2009. The fluo parameter
# is used to change the smoothness of the fluorescence curve and the space between
#  two consecutive measured peaks (droplets).
dropletf <- sim_ddpcr(m = 7, n = 20, times = 5, fluo = list(0.1, 0))
summary(dropletf)
```

```
# droplet dpcr - number of molecules
# Similar to the previous example but with five panels but without and modifications
#  to the peaks.
droplet <- sim_ddpcr(m = 7, n = 20, times = 5)
summary(droplet)

# Visualize the results of dropletf and dropletf
# The curves of dropletf are smoother.
par(mfrow = c(1,2))
plot(dropletf, main = "With fluo parameter", type = "l")
plot(droplet, main = "Without fluo parameter", type = "l")
```

`test_counts` *Test counts*

Description

The test for comparing counts from two or more digital PCR experiments.

Usage

```
test_counts(input, ...)
```

Arguments

`input`	adpcr or dpcr object with with "nm" type.
`...`	additional arguments for `glm` function.

Details

`test_counts` fits General Linear Model (using Poisson `family`) to the counts data from different digital PCR experiments. Comparisions between single experiments utilize Tukey's contrast and multiple t-tests using function `glht`.

Value

an object of class

Note

Mean values of Poisson distribution are derived from General Linear Models. They values will vary depending on input.

Author(s)

Michal Burdukiewicz, Stefan Roediger

Examples

```
adpcr1 <- sim_adpcr(m = 10, n = 765, times = 1000, pos_sums = FALSE, n_panels = 3)
adpcr2 <- sim_adpcr(m = 60, n = 550, times = 1000, pos_sums = FALSE, n_panels = 3)
adpcr3 <- sim_adpcr(m = 10, n = 600, times = 1000, pos_sums = FALSE, n_panels = 3)
two_groups <- test_counts(bind_dpcr(adpcr1, adpcr2))
summary(two_groups)
plot(two_groups)
one_group <- test_counts(bind_dpcr(adpcr1, adpcr3))
summary(one_group)
plot(one_group)
```

`test_panel` *Dispersion Test for Spatial Point Pattern in Array dPCR Based on Quadrat Counts*

Description

The function `test_panel` is a convenient wrapper around `quadrat.test` function. Under optimal conditions, the point pattern of dPCR events (e.g., positive droplet & negative droplets). This function can be used to analyze if the pattern on a planar chip is random. Arrays with non-random patterns should be checked for integrity.

Usage

```
test_panel(X, nx_a, ny_a, nx = 5, ny = 5, alternative = c("two.sided",
  "regular", "clustered"), method = c("Chisq", "MonteCarlo"),
  conditional = TRUE, nsim = 1999)
```

Arguments

`X`	Object of the `adpcr` class containing data from one or more panels.
`nx_a`	Number of columns in a plate.
`ny_a`	Number of rows in a plate.
`nx`	Numbers of quadrats in the x direction.
`ny`	Numbers of quadrats in the y direction.
`alternative`	Character string (partially matched) specifying the alternative hypothesis.
`method`	Character string (partially matched) specifying the test to use: either method="Chisq" for the chi-squared test (the default), or method="MonteCarlo" for a Monte Carlo test.
`conditional`	Logical. Should the Monte Carlo test be conducted conditionally upon the observed number of points of the pattern? Ignored if method="Chisq".
`nsim`	The number of simulated samples to generate when method="MonteCarlo".

Details

This function quick-to-use version of quadrat.test function. It works directly on the objects of adpcr. test_panel performs a test of Complete Spatial Randomness for each plate.

Value

An list of objects of class "htest" with the length equal to the number of plates (minimum 1).

Note

A similar result can be achived by using adpcr2ppp and quadrat.test. See Examples.

Author(s)

Adrian Baddeley, Rolf Turner, Michal Burdukiewcz, Stefan Roediger.

References

http://www.spatstat.org/

See Also

quadrat.test.

Examples

```
many_panels <- sim_adpcr(m = 400, n = 765, times = 1000, pos_sums = FALSE,
                   n_panels = 5)
test_panel(many_panels, nx_a = 45, ny_a = 17)

#test only one plate
test_panel(extract_dpcr(many_panels, 3), nx_a = 45, ny_a = 17)

#do test_panel manually
require(spatstat)
ppp_data <- adpcr2ppp(many_panels, nx_a = 45, ny_a = 17)
lapply(ppp_data, function(single_panel) quadrat.test(single_panel))
```

---

test_peaks *Peak Test*

Description

Detect, separate and count positive peaks, negative peaks and peak-like noise. In addition, function calculates area of the peaks.

Arguments

x	a vector containing the abscissa values (e.g., time, position) OR an object of class adpcr.
y	a vector of fluorescence value.
threshold	a value which defines the peak heights not to consider as peak.
noise_cut	a numeric value between 0 and 1. All data between 0 and noise_cut quantile would be considered noise in the further analysis.
savgol	logical value. If TRUE, Savitzky-Golay smoothing filter is used.
norm	logical value. If TRUE, data is normalised.
filter.q	a vector of two numeric values. The first element represents the quantile of the noise and the second one is the quantile of the negative peaks.

Details

The localization of peaks is determined by the findpeaks function. The area under the peak is calculated by integration of approximating spline.

Value

A list of length 2. The first element is a data frame containing: peak number, peak group (noise, negative, positive), position of the peak maximum, area under the peak, peak width, peak height, position of the peak and time resolution.

The second element contains smoothed data.

Author(s)

Stefan Roediger, Michal Burdukiewicz.

References

Savitzky, A., Golay, M.J.E., 1964. Smoothing and Differentiation of Data by Simplified Least Squares Procedures. Anal. Chem. 36, 1627-1639.

Examples

```
data(many_peaks)
par(mfrow = c(3,1))
plot(many_peaks, type = "l", main = "Noisy raw data")
abline(h = 0.01, col = "red")

tmp.out <- test_peaks(many_peaks[, 1], many_peaks[, 2], threshold = 0.01, noise_cut = 0.1,
                      savgol = TRUE)
plot(tmp.out[["data"]], type = "l", main = "Only smoothed")
abline(h = 0.01, col = "red")
abline(v = many_peaks[tmp.out[["peaks"]][, 3], 1], lty = "dashed")

tmp.out <- test_peaks(many_peaks[, 1], many_peaks[, 2], threshold = 0.01, noise_cut = 0.1,
                      savgol = TRUE, norm = TRUE)
```

```
plot(tmp.out[["data"]], type = "l", main = "Smoothed and peaks detected")
abline(v = many_peaks[tmp.out[["peaks"]][, 3], 1], lty = "dashed")
for(i in 1:nrow(tmp.out$peaks)) {
  if(tmp.out$peaks[i, 2] == 1) {col = 1}
  if(tmp.out$peaks[i, 2] == 2) {col = 2}
  if(tmp.out$peaks[i, 2] == 3) {col = 3}
  points(tmp.out$peaks[i, 7], tmp.out$peaks[i, 6], col = col, pch = 19)
}

positive <- sum(tmp.out$peaks[, 2] == 3)
negative <- sum(tmp.out$peaks[, 2] == 2)
total <- positive + negative
```

`test_ratio` *Rate ratio test*

Description

The test for comparing ratio of two Poisson means: $r = \frac{\lambda_1}{\lambda_2}$.

Usage

```
test_ratio(dpcr1, dpcr2, alternative = c("two.sided", "less", "greater"),
  conf.level = 0.95)
```

Arguments

`dpcr1`	a (non-empty) numeric vector of data values of length 2 or more or an object of class `adpcr` or `ddpcr`. See Details.
`dpcr2`	a (non-empty) numeric vector of data values of length 2 or more or an object of class `adpcr` or `ddpcr`. See Details.
`alternative`	alternative hypothesis, must be one of: `two.sided`, `greater` or `less`.
`conf.level`	confidence level for the returned confidence interval.

Details

Objects `dpcr1` and `dpcr2` can be:

1. numeric vectors of length 2. The first element is assumed to be number of positive partitions and the second one to be the total number of partitions.
2. numeric vectors of length greater than 2. The length of vector is assumed to represent total number of partitions. Every element of the vector with value bigger than 0 is assumed to be a positive partitions.
3. `adpcr` objects with type `tp` (total number of positive wells in panel) or `nm` (number of molecules per partition). `ddpcr` objects with type `tp` (total number of positive droplets) or `nm` (number of molecules per droplet).

Both `dpcr1` and `dpcr2` must have the same class. See Examples.

The `ratio_test` is a wrapper around rateratio.test function with custom input and output tailored specifically for digital PCR experiments.

Value

An object of class 'htest' containing the following components:

`p.value`	the p-value of the test
`estimate`	a vector with the means (lambdas) of both experiments and their ratio
`conf.int`	confidence interval for ratio between two experiments.
`alternative`	type of alternative hypothesis
`method`	description of method
`data.name`	description of data

Author(s)

Michael Fay, Michal Burdukiewicz, Stefan Roediger

References

Fay M.P. *Two-sided exact tests and matching confidence intervals for discrete data* R Journal 2 (1), 2010.

See Also

See also `poisson.test`.

Examples

```
# Input values are numeric vectors representing dPCR experiments
x1 <- rpois(765, 1.1)
x2 <- rpois(765, 1.1)
test_ratio(x1, x2)

# Input values represent only number of positive partitions and total
# partitions
x3 <- sum(rpois(765, 1.1) > 0)
x4 <- sum(rpois(765, 1.1) > 0)
test_ratio(c(x3, 765), c(x4, 765))

# It is possible to mix different types of input as long as they have
# the same class
test_ratio(c(x3, 765), x1)

# The same is true for adpcr and dpcr objects.
x5 <- sim_adpcr(400, 1600, 100, pos_sums = TRUE, n_panels = 1)
x6 <- sim_adpcr(400, 1600, 100, pos_sums = FALSE, n_panels = 1)
test_ratio(x5, x6)

x7 <- sim_ddpcr(400, 1600, 100, pos_sums = TRUE, n_exp = 1)
x8 <- sim_ddpcr(400, 1600, 100, pos_sums = FALSE, n_exp = 1)
test_ratio(x7, x8)
```

Index