Package ‘dpcR’

February 4, 2019

Type Package
Title Digital PCR Analysis
Version 0.5
LazyData true
Date 2019-02-01
Description Analysis, visualisation and simulation of digital polymerase chain reaction (dPCR) (Burdukiewicz et al. (2016) <doi:10.1016/j.bdq.2016.06.004>). Supports data formats of commercial systems (Bio-Rad QX100 and QX200; Fluidigm BioMark) and other systems.
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, evd, dgoaf, multcomp, qpcR, pracma, rateratio.test, readxl, signal, shiny, spatstat
Suggests digest, DT, ggplot2, knitr, markdown, rhandsontable, rmarkdown, shinythemes, xtable
VignetteBuilder knitr
NeedsCompilation no
Repository CRAN
Encoding UTF-8
RoxygenNote 6.1.1
Maintainer Michal Burdukiewicz <michalburdukiewicz@gmail.com>
Collate 'AUCtest.R' 'BioradCNV.R' 'White.R' 'adpcr2panel.R'
      'adpcr2ppp.R' 'binarize.R' 'classes.R' 'bind_dpcr.R' 'bioamp.R'
      'calc_breaks.R' 'calc_coordinates.R' 'calc_lambda.R'
      'compare_dens.R' 'count_test-class.R' 'create_dpcr.R'
      'dPCRmethyl.R' 'dpcRquant.R' 'df2dpcr.R' 'dpcR-package.R'
      'dpcReport_gui.R' 'dpcr2df.R' 'dpcr_calculator.R'
      'dpcr_density.R' 'dpcr_density_gui.R' 'dpcr_density_table.R'

1
R topics documented:

'extract_run.R' 'fit_adpcr.R' 'fl.R' 'get_k_n.R' 'hsm.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' 'qdpcr.R'
'qpcr2pp.R' 'qpcr_analyser.R' 'read_amp.R' 'read_dpcr.R'
'rename_dpcr.R' 'rtadpcr.R' 'safe_efficiency.R' 'show_dpcr.R'
'sim_adpcr.R' 'sim_ddpcr_bkm.R' 'sim_dpcr.R' 'simulations.R'
'six_panels.R' 'summary_dpcr.R' 'test_counts.R'
test_counts_gui.R' 'test_panel.R' 'test_peaks.R'
test_pooled.R' 'valid_amp.R' 'y_val_conf.R'

Author  Michal Burdukiewicz [cre, aut]
(<https://orcid.org/0000-0001-8926-582X>),
Stefan Roediger [aut] (<https://orcid.org/0000-0002-1441-6512>),
Bart Jacobs [aut],
Piotr Sobczyk [ctb],
Andrei-Nikolai Spiess [ctb]

Date/Publication  2019-02-03 23:13:49 UTC

R topics documented:
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- and heterozygosity, as well as 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 relating it to the total number of compartments by means of Poisson statistics which enables an absolute quantification.
Details

The package includes plot functions, summary functions, data sets and simulations for dPCR and customizable GUI for droplet digital PCRs and array-based digital PCRs. We aim to include all statistical approaches published in peer-review literature and additional selected sources of expertise currently available. We intend to make these methods available to the scientific community in an open and cross-platform environment. Using the naming convention derived from the MIQE Guidelines for digital PCR, we hope to become a reference to a unified nomenclature in dpcR.

The package is primarily targeted at researchers who wish 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 to better learn about digital PCR concepts and data interpretation.

Author(s)

Michal Burdukiewicz, Stefan Roediger.

Maintainer: Michal Burdukiewicz <michalburdukiewicz@gmail.com>

References


See Also


Examples

```r
adpcr <- sim_adpcr(m = 400, n = 765, times = 20, pos_sums = FALSE, n_panels = 1)
plot_panel(adpcr, 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". Inherits from dpcr.
Details
For more in-depth explanation of digital PCR data structure, see dpcr.

Slots

col_names "character" vector naming columns in the array.
row_names "character" vector naming rows in the array.
row_id "integer" vector providing row indices of all runs.
col_id "integer" vector providing column indices of all runs.
panel_id "factor" naming the panel to which experiment belong.

Author(s)
Michal Burdukiewicz.

See Also
Data management: adpcr2panel, bind_dpcr, extract_run.
Plotting: plot_panel.
Tests: test_panel.
Simulation: sim_adpcr.
Real-time array digital PCR: rtadpcr.
Droplet digital PCR: dpcr.

Examples

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

adpcr2panel  Convert adpcr object to array

Description
Converts adpcr object into the list of array-like matrices.

Usage
adpcr2panel(input, breaks = FALSE)
Arguments

input  object of the \code{adpcr} class.
breaks if \code{TRUE}, the data is divided into intervals.

Value

A named list of length equal to the number of arrays in the \code{input}. Each element is a single array in matrix-like form, where dimensions are set exactly as in case of the real plate. Names of the list corresponds to the names of assays ("tnp" data) or runs (any other type of \code{adpcr} data). The matrices contain values from array, either integers (when \code{use_break} is \code{FALSE}) or characters (when \code{use_break} is \code{TRUE}).

Author(s)

Michal Burdukiewicz.

Examples

```r
# generate data
ttest <- sim_adpcr(m = 400, n = 765, times = 20, pos_sums = FALSE,
                   n_panels = 3)
# convert object into three arrays
arrays <- adpcr2panel(ttest)
length(arrays)
# print an array
arrays[[1]]
```

---

\textbf{adpcr2ppp} \hfill \textit{Convert adpcr to ppp}

Description

Converts \code{adpcr} object to the list of \code{ppp.objects}.

Usage

\code{adpcr2ppp(input, marks = TRUE, plot = FALSE)}

Arguments

- \texttt{input} Object of the \code{adpcr} class containing data from one or more panels.
- \texttt{marks} If \code{TRUE}, marks values for non-empty partitions. See \code{ppp} for more in-depth description.
- \texttt{plot} If \code{TRUE}, array is plotted.
**binarize**

**Binarize digital PCR data**

**Description**

Transforms multinomial (number of molecules per partition) or continuous (fluorescence) digital PCR data to binary (positive/negative partition) format.

**Details**

Each array 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 arrays (minimum 1).

**Author(s)**

Michal Burdukiewcz, Stefan Roediger.

**See Also**

`ppp.object`, `ppp`.

**Examples**

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

# Convert all arrays to ppp objects
adpcr2ppp(many_panels)

# Convert all arrays to ppp objects and get third plate
third_plate <- adpcr2ppp(many_panels)[[3]]

# Convert only third plate to ppp object
third_plate2 <- adpcr2ppp(extract_run(many_panels, 3))

# 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]])
```
Usage

bind_dpcr(input)

Arguments

- **input**: object of the class `adpcr` or `dpcr` with one of the following types: "ct", "fluo" or "nm".

Value

- object of the class `adpcr` or `dpcr` (depending on `input`) with type "np".

Author(s)

Michal Burdukiewicz.

Examples

```r
# adpcr object
rand_array <- sim_adpcr(200, 300, 100, pos_sums = FALSE, n_panels = 1)
bind_dpcr(rand_array)

# dpcr object
rand_droplets <- sim_dpcr(200, 300, 100, pos_sums = FALSE, n_exp = 1)
bind_dpcr(rand_droplets)
```

Description

A convenient 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 `dpcr` or a list.
- **...**: objects of class `adpcr` or `dpcr`. See Details. If `input` is a list, ignored.
Details

In case of adpcr or dpcr objects, bind_dpcr 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 dpcr, 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_run

Examples

```r
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_dpcr(m = 7, n = 20, times = 5, n_exp = 2)
bigger_droplet <- sim_dpcr(m = 15, n = 25, times = 5, n_exp = 4)
biggest_droplet <- sim_dpcr(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 Bio-Rad droplet digital PCR experiment

Description

bioamp is a function to plot and analyze the amplitude data of a Bio-Rad 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 channel (x-axis).
amp_y          is the second amplitude channel (y-axis).
cluster        are the clusters of the plot. The number indicates the column of a table, which contains the cluster information.
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["D01"], main = "Well D01", pch = 19)
bioamp(data = pds_raw["D02"], main = "Well D02", pch = 19)
par(mfrow = c(1,1))

BioradCNV

Copy number variation experiment

Description

Copy Number Variation of MYC Gene for seven patients measured using QX200 (Bio-Rad)

Format

An object of class adpcr with "tnp" type containing four runs from seven experiments (four runs per each experiment).
**calc_coordinates**

**Author(s)**

Robert M. Dorazio, Margaret E. Hunter.

**Source**


**Examples**

```r
data(BioradCNV)
summary(BioradCNV)
```

**Description**

Calculates coordinates of points on plot to represent a digital PCR array.

**Usage**

```r
calc_coordinates(array, half)
```

**Arguments**

- `array` A single array, as generated by `adpcr2panel`.
- `half` If `left` or `right`, every well is represented only by the adequate half of the rectangle.

**Value**

Returns two sets of coordinates of each microfluidic well: `coords` is a list of coordinates suitable for usage with functions from `graphics` package. The second element is a data frame of coordinates useful for users utilizing `ggplot2` package.

**Author(s)**

Michal Burdukiewicz, Stefan Roediger.

**See Also**

- `plot_panel` - plots `adpcr` data. `adpcr2panel` - converts `adpcr` object to arrays.
The function `compare_dens` is used to plot empirical and theoretical density of the result of a digital PCR experiment. Its usage is as follows:

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

**Arguments**

- `input`: object of class `dpcr` containing only one run.
- `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**

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

---

`count_test` is a class that objects of this class are created by `test_counts`. Objects of this class are created by `test_counts`. 

**Description**

Objects of this class are created by `test_counts`. 

---
count_test

Usage

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

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

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

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

Arguments

- `object` of class `count_test`
- `x` object of class `count_test`
- `aggregate` logical, if TRUE experiments are aggregated according to their group.
- `nice` logical, if TRUE a more aesthetically pleasing (but harder to customize) version of the plot is created.

Details

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

Methods (by generic)

- `summary`: Summary statistics of assigned groups.
- `coef`: Extract coefficients of groups.
- `show`: Print both `group_coef` and `test_res`.
- `plot`: plots mean number of molecules per partition and its confidence intervals.

Slots

- `group_coef` "data.frame" containing experiments, groups to which they belong and calculated values of rate (lambda).
- `test_res` "matrix" containing result of multiple comparisions t-test.
- `model` "character" name of GLM used to compare experiments.

Author(s)

Michal Burdukiewicz.

See Also

test_counts.
create_dpcr

Create dpcr object

Description

Creates adpcr and dpcr objects from data.

Usage

create_dpcr(data, n, exper = "Experiment 1", replicate = NULL,
            assay = "Unknown", type, v = 1, uv = 0, threshold = NULL, adpcr,
            col_names = NULL, row_names = NULL, panel_id = NULL)

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.
exper The id of experiments.
replicate The id of technical replicates.
assay The name or id of assays.
type Object of class "character" defining type of data. Could be "nM" (number
of molecules per partition), "tnp" (total number of positive wells in the panel),
"flu" (fluorescence), "np" (status (positive (1) or negative(0)) of each droplet)
or "ct" (threshold cycle).
v The volume of partitions [nL].
uv The volume uncertainty of partitions [nl].
threshold numeric value giving the threshold above which droplet is counted as positive.
Ignored if adpcr is TRUE.
adpcr logical. If TRUE, function creates adpcr object. If FALSE, function creates
dpcr object.
col_names character vector of column names in array. Ignored if not adcpr.
row_names character vector of row names in array. Ignored if not adcpr.
panel_id factor vector of panel IDs (or names). Ignored if not adcpr.

Details

This constructor function assists in creation of objects used by other functions of the package. It is
also responsible for checking the correctness of arguments.
A warning is prompted whenever any of arguments is converted to other type.

Value

An adpcr or dpcr object.
Note

create_dpcr is preferred to calling directly new. Currently only end-point measurements are supported.

Author(s)

Michal Burdukiewicz, Stefan Roediger.

See Also

Streamlined, but more limited version: df2dpcr

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)

ddpcRquant

Quantify droplets

Description

Cluster raw data from QX100 and QX200 systems.

Usage

ddpcRquant(path, threshold.int = 0.9995, reps = 10, blocks = 150,
  threshold.manual = NULL)

Arguments

  path character path to directory with raw data.
  threshold.int numeric probability of the threshold quantile.
  reps numeric vector representing the number of replications.
  blocks numeric vector representing the number of blocks.
threshold.manual
   if numeric, the value is used as the threshold. If NULL, the threshold is calculated automatically.

Value
   dpcr object.

Note
   This function is a modification of code was implemented using the code found on http://www.ddpcrquant.ugent.be/.

Author(s)
   Wim Trypsteen, Matthijs Vynck, Jan De Neve, Pawel Bonczkowski, Maja Kiselinova, Eva Malatinkova, Karen Vervisch, Olivier Thas, Linos Vandekerckhove, Ward De Spiegelaere.

References

---

df2dpcr

**Convert data.frame to dpcr object**

**Description**

Converts data.frame object to to adpcr or dpcr object. The resulting object will have "trp" type.

**Usage**

df2dpcr(df)

**Arguments**

df data frame with specified column names. See Details.

**Details**

The data frame must have REDF structure. It means that data must contain following columns with exactly specified names:

- **experiment** names of experiments
- **replicate** indices of replicates
- **assay** names of assays
**dpcr-class**

- **k** number of positive partitions
- **n** total number of partitions
- **v** volume of partition (nL)
- **uv** uncertainty of partition’s volume (nL)

**threshold** partitions with k equal or higher than threshold are treated as positive.

There are also one optional column:

- **panel_id** indices of panels

If the additional column is present, the resulting object has **adpcr** type.

**Value**

An object of **adpcr** or **dpcr** type, depends on the presence of additional column with panel indices (see Details).

**Author(s)**

Michal Burdukiewcz, Stefan Roediger

**See Also**

Flexibly create **dpcr** objects: **create_dpcr** Inverse function: **dpcr2df**

**Examples**

```r
dat <- data.frame(experiment = factor(rep(paste0("Experiment", 1L:2), 3)),
                   replicate = c(1, 1, 2, 2, 3, 3),
                   assay = "Assay1",
                   k = c(55, 121, 43, 150, 70, 131),
                   n = 765,
                   v = 1,
                   uv = 0)
df2dpcr(dat)
```

---

**dpcr-class**

A class containing results of any digital PCR experiment. Type of data in all columns is specified in slot "type".

---

**Class "dpcr" - general digital PCR**
Details

Possible type values of dpcr objects:

1. "ct": cycle threshold of each partition,
2. "fluo": fluorescence of each partition,
3. "nm": number of molecules in each partition,
4. "np": status (positive (1) or negative(0)) of each partition,
5. "tnp": total number of positive partitions in the run (single value per each run, not per partition).

Digital PCR data is always a matrix, where columns and rows represent respectively runs and data points. For example, matrix with 2 columns and 765 rows means two runs with 765 data points each. In case of "tnp" data, each run is represented by only one measurement, the count of all positive partitions.

The number of partitions is defined in slot n. In the previous example, two runs have 765 data points, but they can have less detected partitions (for example some reads may be not available). In this case, the data point will have value NA.

The structure of dpcr class is described more deeply in the vignette.

Slots

.Data matrix data from digital PCR experiments. See Details.

n integer equal to the number of partitions in each run.

exper factor the id or name of experiments.

replicate factor the id or name of replicates.

assay factor the id or name of the assay.

v "numeric" volume of the partition [nL].

uv "numeric" uncertainty of the volume of the partition [nL].

threshold "numeric" value specifying the threshold. Partition with the value equal or bigger than threshold are considered positive.

type Object of class "character" defining type of data. See Details.

Note

This class represent the most general droplet-based digital PCR. In more specific cases, the user is directed to other classes: adpcr, where results can be placed over a plate, qdpcr where digital assay is based on multiple qPCR experiments and rtadpcr, where data points represent the status of partitions measured in the real time.

Author(s)

Michal Burdukiewicz.
Examples

```r
dpcr_fluo <- sim_dpcr(m = 10, n = 20, times = 5, fluo = list(0.1, 0))
plot(dpcr_fluo)
dpcr <- sim_dpcr(m = 10, n = 20, times = 5)
```

---

**dpcr2df-methods**  
*Convert dpcr object to data frame*

**Description**

Converts \texttt{adpcr} or \texttt{dpcr} object to \texttt{data.frame} object.

**Usage**

```r
dpcr2df(input)
```

**Arguments**

- `input` \texttt{adpcr} or \texttt{dpcr} object.

**Value**

data frame with 5 (if input was \texttt{dpcr}) or 8 columns (if input was \texttt{adpcr}).

**Author(s)**

Michal Burdukiewcz, Stefan Roediger

**See Also**

Inverse function: \texttt{df2dpcr}

**Examples**

```r
dpcr2df(six_panels)
```
**dpcReport**

*Digital PCR Report Graphical User Interface*

### Description

Launches graphical user interface that generates reports from digital PCR data.

### Usage

```r
dpcReport()
```

### Warning

Any ad-blocking software may cause malfunctions.

### Author(s)

Michal Burdukiewicz, Stefan Roediger.

---

**dPCRmethyl**

*Methylated human gDNA*

### Description

Results of dPCR experiment: serial dilutions (0.25, 0.50, 0.75, 1.00) of methylated human gDNA. For each concentration level, three samples were measured using QX100 (Bio-Rad).

### Format

A `adpcr` object

### Author(s)

Mario Menschikowski, Stefan Roediger, Michal Burdukiewcz

### Source

Mario Menschikowski group / Institute of Clinical Chemistry and Laboratory Medicine, TU-Dresden, Dresden, Germany

### Examples

```r
summary(dPCRmethyl)
```
dpcr_density  

**Calculate Density of Single dPCR Run**

**Description**

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**

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

**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`: Method for calculating the confidence interval. 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.
- `bars`: plot on density plot bars for discrete values of lambda.
- `...`: Additional arguments send to `plot` function.

**Value**

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

**Author(s)**

Michal Burdukiewicz, Stefan Roediger.

**References**

See Also

Computation of confidence intervals: binom.confint,
The browser-based graphical user interface for this function: dpcr_density_gui.

Examples

```r
# 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 = 0.95)
dpcr_density(k = 25, n = 55, average = TRUE, methods = "wilson",
             conf.level = -0.95)
par(mfrow = c(1,1))

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

---

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 cause malfunctions.

Author(s)

Michal Burdukiewicz, Stefan Roediger.

See Also

dpcr_density.
dpcr_density_table Calculate Density of Multiple dPCR runs

Description
Calculates the density of the number of positive molecules or the average number of molecules per partition of dpcr objects.

Usage
dpcr_density_table(input, average = FALSE, methods = "wilson", conf.level = 0.95)

Arguments
input an object of class dpcr.
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 Method for calculating the confidence interval. 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.

Value
A list (with the length equal to the number of runs in input) of data frames containing densities and borders of confidence intervals.

Author(s)
Michal Burdukiewicz, Stefan Roediger.

See Also
dpcr_density for easy analysis and plots of single runs.

Examples
dens <- dpcr_density_table(six_panels)

# create plot using ggplot2
library(ggplot2)
ggplot(dens[["Experiment2.2"]], aes(x = x, y = y)) + geom_line() +
extract_dpcr

Description

Extract all runs belonging to specific assay or experiment(s) from a code_pcr object while preserving all other attributes.

Usage

extract_dpcr(input, id_exper = NULL, id_assay = NULL)

Arguments

- **input**: object of the class adpcr or dpcr.
- **id_exper**: vector of indices or names of experiments. Must be NULL if id_assay is specified.
- **id_assay**: vector of indices or names of assays. Must be NULL if id_exper is specified.

Value

The object of the input’s class (adpcr or dpcr).

Note

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

Author(s)

Michał Burdukiewicz.

See Also

Extract run(s): extract_run.
**extract_run**

**Examples**

```r
#extract using only experiment's ID
event_extract_dpcr(six_panels, id_exper = 1)

#extract using assay name
extract_dpcr(six_panels, id_assay = "MYC")

#extract using multiple names
extract_dpcr(six_panels, id_exper = c("Experiment1", "Experiment2"))
```

**Description**

Extract runs from a codedpcr object while preserving all other attributes.

**Usage**

`extract_run(input, id)`

**Arguments**

- `input`: object of the class `adpcr` or `dpcr`.
- `id`: vector of indices or names of runs.

**Details**

The `extract_run` function allows to choose one or more panels from an object of the `adpcr` or `dpcr` 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 `dpcr`).

**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 Extract multiple runs belonging to an experiment of assay: extract_dpcr

Examples

```r
#sample extracting
pans <- sim_adpcr(10, 40, 1000, pos_sums = FALSE, n_panels = 50)
single_panel <- extract_run(pans, 5)
random_three <- extract_run(pans, sample.int(nrow(pans), 3))
all_but_one <- extract_run(pans, -5)

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

---

**limit_cq**

*Limit Cy0 values*

**Description**

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**

```r
limit_cq(data, cyc = 1, fluo = NULL, Cq_range = c(1, max(data[cyc])),
         model = 15, SDM = TRUE, pb = FALSE)
```

**Arguments**

data: a data frame 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 '15' (see pcrfit).
SDM: if TRUE, Cq is approximated by the second derivative method. If FALSE, Cy0 method is used instead.
pb: if TRUE, progress bar is shown.
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


See Also

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

Cy0 method: `mselect`, `efficiency`.

Examples

```r
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 = 15)
# Same as above, but without Cq.range
no_range <- limit_cq(data = test, cyc = 1, fluo = NULL, model = 15)
```
```r
# Same as above, but only three columns
no_range234 <- limit_cq(data = test, cyc = 1, fluo = c(2:4), model = 15)
```

---

**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**

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

---

**moments-methods**

*Calculate Moments of Poisson Distribution*

---

**Description**

Computes moments of a Poisson distribution. The calculations are based on values of positive and total partitions or the theoretical lambda value.

**Usage**

```r
moments(input)
```

**Arguments**

- **input** a vector with two elements (the first element is treated as a number of positive partitions and the second as a number of total partitions) or a matrix with two columns (first columns contains numbers of positive partitions and the second total numbers of total partitions) or an object of class `dpcr`.
Value
A data frame with four columns: name of the experiment, name of the replicate, method of computation (theoretical or empirical), name of the moment and the value of the moment. The theoretical moments are computed using the lambda value and the empirical using the sample values.

Note
Four first moments of a Poisson distribution.
Mean: $\lambda$.
Variance: $\lambda$.
Skewness: $\sqrt{\lambda}$.
Kurtosis: $\frac{1}{\lambda}$.

Author(s)
Michal Burdukiewicz.

Examples
C moments for QPP positive partitions of WVU total partitions
momentsHcHQPPL WVUII
C calculate moments for an array digital PCR
moments(six_panels)

| 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)
Plasmid dilution series results

Description

These are the results data from the pds_raw data as calculated by the Bio-Rad 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 10^3 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
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

**Examples**

```r
summary(extract_run(read_QX100(pds), 1L:10))
```

---

**pds_raw  
Plasmid dilution series raw data**

---

**Description**

Subset of raw data from the `pds_raw` data set as measured by the Bio-Rad QX100 Droplet Digital PCR System.

**Format**

A list of 3 data frames.

<table>
<thead>
<tr>
<th>Well</th>
<th>ExpType</th>
<th>Experiment</th>
<th>Sample + Dilution step</th>
<th>TypeAssay</th>
<th>Assay</th>
</tr>
</thead>
<tbody>
<tr>
<td>D01</td>
<td>Absolute Quantification</td>
<td>ABS</td>
<td>gDNA + P 10^4</td>
<td>Ch1Unknown</td>
<td>ileS</td>
</tr>
<tr>
<td>D02</td>
<td>Absolute Quantification</td>
<td>ABS</td>
<td>gDNA + P 10^2</td>
<td>Ch1Unknown</td>
<td>ileS</td>
</tr>
<tr>
<td>C04</td>
<td>Absolute Quantification</td>
<td>ABS</td>
<td>gDNA</td>
<td>Ch1Unknown</td>
<td>ileS</td>
</tr>
</tbody>
</table>

**Details**

The results can be calculated as by the Bio-Rad QX100 Droplet Digital PCR System are to be found in `pds`.  

Setup: Duplex assay with a 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).

# Setup: Duplex assay with a 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).

The full data are located at https://github.com/michbur/dpcr_data as QX100_rawdata.zip.

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

Examples

```r
#str(pds_raw)
bioamp(data = pds_raw["D02"], main = "Well D02", pch = 19)
```

plot.qdpcr

Plot qdpcr 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 qdpcr 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 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

qdpcr

Examples

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

plot(qpcr2pp(data = test, cyc = 1, fluo = NULL, model = 15, delta = 5), rug = TRUE)
```

---

**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**

```r
plot_panel(input, col = "red", legend = TRUE, half = "none", plot = TRUE, ...)
```

**Arguments**

- `input` object of the `adpcr` class.
- `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.
- `plot` "logical", if FALSE, only plot data is returned invisibly.
- `...` Arguments to be passed to plot function.
**plot_panel**  

### Details

Currently, only objects containing `tnp` data can be plotted as a whole. For any other type of the `adpcr` data, only just one column of data (one panel) can be plotted at the same time (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

Invisibly returns two sets of coordinates of each microfluidic well as per `calc_coordinates`: `coords` is a list of coordinates suitable for usage with functions from `graphics` package. The second element is a data frame of coordinates useful for users utilizing `ggplot2` package.

### Author(s)

Michal Burdukiewicz, Stefan Roediger.

### See Also

- `extract_run` - extract experiments.  
- `adpcr2panel` - convert `adpcr` object to arrays.

### Examples

```r
# 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)  

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

# 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_run(ttest2, i), legend = FALSE,  
  main = paste("Panel", LETTERS[i], sep = " ")))  
par(mfcol = c(1, 1))  

# two different channels  
plot_panel(extract_run(ttest2, 1), legend = FALSE,  
  half = "left")  
par(new = TRUE)  
plot_panel(extract_run(ttest2, 2), col = "blue",  
```

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 dpcr.
fam Amplitudes of the FAM channel - object of class dpcr.
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

Examples

# Generate an amplitude plot for the first fluorescence channel (e.g., FAM)
fluos1 <- sim_dpcr(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_dpcr(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_dpcr(m = 16, n = 30, times = 100, pos_sums = FALSE, n_exp = 2,
                      fluo = list(0.1, 0))
plot_vic_fam(extract_run(fluos_both, 1), extract_run(fluos_both, 2))

description
An object representing digital PCR reaction depicted as Poisson process. Inherits from dpcr.

Slots
list("mu")  "numeric" of the expected number of events in a defined interval.
list("C0")  "numeric" the occurrence in a defined interval.
list("CT")  "numeric" value of the "average time" between the occurrence of a positive reaction
             and another positive reaction.
Author(s)

Stefan Roediger, Michal Burdukiewicz.

See Also

plot.qdpcr.

qpcr2pp

qPCR to Poisson Process

Description

Describes qPCR as Poisson process.

Usage

qpcr2pp(data, cyc = 1, fluo = NULL, Cq_range = c(min(data[cyc]) + 6, max(data[cyc]) - 6), model = "l5", SDM = TRUE, NuEvents = 1, delta = 1, exper = "qPCR", replicate = 1, assay = "Unknown", type = "np")

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.
NuEvents "number of expected events" within a time frame (interval).
delta difference "time (cycles) points" e.g., Cycle 18 and 25.
exper The id of experiments.
replicate The id of technical replicates.
assay The name or id of assays.
type object of class "character" defining type of data. Could be "np" (status (positive (1) or negative(0)) of each droplet) or "ct" (threshold cycle).
Details

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.

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 `qdpcr` class.

Author(s)

Stefan Roediger, Michal Burdukiewicz.

Examples

```r
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
qpcrpp <- qpcr2pp(data = test, cyc = 1, fluo = NULL, Cq_range = c(20, 30),
                   model = 15, delta = 5)
```
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 'l5' (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_analyser` is a function 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 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 function, 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


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)
**read_amp**

*Read digital PCR amplitude raw data*

---

**Description**

Reads digital PCR amplitude data.

**Usage**

```r
read_amp(input, ext = NULL)
```

**Arguments**

- `input`: name of the input file (character) or input object (data.frame).
- `ext`: extension of the file ().

**Details**

The amplitude data means a compressed directory of amplification.

**Value**

An object of `adpcr`.

**Author(s)**

Michal Burdukiewcz, Stefan Roediger

---

**read_BioMark**

*Read BioMark*

---

**Description**

Reads digital PCR data from the BioMark (Fluidigm).

**Usage**

```r
read_BioMark(input, ext = NULL, detailed = FALSE)
```

**Arguments**

- `input`: name of the input file (character) or input object (data.frame).
- `ext`: extension of the file ().
- `detailed`: logical, if TRUE, the input file is processed as if it was 'Detailed Table Results'. In the other case, the expected input file structure is 'Summary Table Results'.
**read_dpcr**

**Value**
An object of `adpcr` class.

**Author(s)**
Michał Burdukiewcz, Stefan Roediger

**References**

**See Also**
See `read_dpcr` for detailed description of input files.

---

**read_dpcr**

**Description**
Reads digital PCR data in various formats.

**Usage**
```r
read_dpcr(input, format, ext = NULL, ...)
```

**Arguments**
- `input`: name of the input file (character) or input object (data.frame).
- `format`: of the file, for example: "raw", "QX100", "BioMark", "amp" (raw amplitude compressed using .zip).
- `ext`: extension of the file ().
- `...`: additional parameters for the appropriate function. For example, if format has value "raw", the additional parameter must be `adpcr`.

**Details**
Input files may have .csv, .xls or .xlsx extension. In case of Excel files with multiple sheets, only the first sheet will be analyzed.

**Value**
Always an object of `adpcr` or `dpcr` type.
Author(s)
Michal Burdukiewcz, Stefan Roediger

See Also

Description
Reads digital PCR data from the QX100 Droplet Digital PCR System (Bio-Rad).

Usage
read_QX100(input, ext = NULL)

Arguments
input name of the input file (character) or input object (data.frame).
ext extension of the file ().

Value
An object of adpcr class.

Note
The volume and its uncertainty are taken from the literature (see references).

Author(s)
Michal Burdukiewcz, Stefan Roediger

References

See Also
See read_dpcr for detailed description of input files.
Example of QX100 data: pds.
**Description**

Reads digital PCR data from the QX200 Droplet Digital PCR System (Bio-Rad).

**Usage**

`read_QX200(input, ext = NULL)`

**Arguments**

- **input**: name of the input file (character) or input object (data.frame).
- **ext**: extension of the file (character).

**Value**

An object of `adpcr` class.

**Note**

The volume and its uncertainty are taken from the literature (see references).

**Author(s)**

Michal Burdukiewcz, Stefan Roediger

**Source**


**References**


**See Also**

See `read_dpcr` for detailed description of input files.
**Description**

Reads REDF (Raw Exchange Digital PCR format) data.

**Usage**

```r
read_redf(input, ext = NULL)
```

**Arguments**

- `input` name of the input file (character) or input object (data.frame).
- `ext` extension of the file ().

**Details**

REDF (Raw Exchange Digital PCR format) data is preferably a .csv file with following columns:

- `experiment` names of experiments
- `replicate` indices of replicates
- `assay` names of assays
- `k` number of positive partitions
- `n` total number of partitions
- `v` volume of partition (nL)
- `uv` uncertainty of partition’s volume (nL)
- `threshold` partitions with k equal or higher than threshold are treated as positive.
- `panel_id` indices of panels

Column `panel_id` should be specified only in case of array-based dPCR.

**Value**

An object of `adpcr` or `dpcr` type, depends on the value of `adpcr` parameter.

**Author(s)**

Michal Burdukiewcz, Stefan Roediger
rename_dpcr

rename_dpcr  Rename object

Description
Renames objects of class adpcr or dpcr.

Usage
rename_dpcr(x, exper = NULL, replicate = NULL, assay = NULL)

Arguments
x  an adpcr or dpcr object.
exper  a vector of new experiments' names. If NULL, experiments' names are not changed.
replicate  a vector of new replicates' ids. If NULL, replicates' names are not changed.
assay  a vector of new assays' names. If NULL, assays' names are not changed.

Details
The valid exper, replicate and assay names are factors. For the sake of convenience, this function converts other types to factors if it is possible.

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: `dpcr`.

Examples

```r
#none
```

Description
Expands function `show` allowing showing objects of the class `adpcr` or `dpcr`.

Arguments

- `object`: an object of class `dpcr`.

Author(s)
Michal Burdukiewicz.

Examples

```r
#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_dpcr(7, 20, times = 5, fluo = list(0.1, 0))
show(dropletf)

#droplet dpcr - number of molecules
droplet <- sim_dpcr(7, 20, times = 5)
show(droplet)
```
**sim_adpcr**  
*Simulate Array Digital PCR*

**Description**

A function that simulates results of an array digital PCR.

**Usage**

```r
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 and their approximate number in a single chamber.

The function contains two implementations of the array digital PCR simulation. First one was described in Dube et 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 \( n_{\text{panels}} \) columns. Each column represents one plate. The type of such simulation would be "nm". If the pos_sums argument has value TRUE, the function returns a matrix with one row and \( n_{\text{panels}} \) columns. Each column contains the total number of positive chambers in each plate and type of simulation would be set as "tnp".

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

Author(s)

Michal Burdukiewicz.

References


See Also

sim_dpcr.

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 repli <- 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))
`sim_dpcr`  

Simulate Droplet Digital PCR

Description

A function that simulates results of a droplet digital PCR.

Usage

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

Arguments

- **m**: the total number of template molecules used in the experiment. Must be a positive integer.
- **n**: the number of droplets per experiment. 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

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 \( m \times \text{times} \) molecules between \( n \times \text{times} \) 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. The 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 "tnp".

In each case the value is an object of the \texttt{dpcr} 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

\texttt{sim_adpcr}.

Examples

```r
# simulate fluorescence data
tmp_VIC <- sim_dpcr(m = 7, n = 20, times = 5, fluo = list(0.1, 0))
tmp_FAM <- sim_dpcr(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_dpcr(m = 7, n = 20, times = 5, n_exp = 5))
```

Description

Simulated data from array-based digital PCR experiment (see \texttt{sim_adpcr}).

Format

An object of class \texttt{adpcr} containing six runs from three experiments (two runs per each experiment).
Examples

# code below was used to create six_panels data set
## Not run:
set.seed(1944)
adpcr1 <- sim_adpcr(m = 10, n = 765, times = 10000, pos_sums = FALSE, n_panels = 2)
adpcr2 <- sim_adpcr(m = 40, n = 765, times = 10000, pos_sums = FALSE, n_panels = 2)
adpcr2 <- rename_dpcr(adpcr2, exper = “Experiment2“)
adpcr3 <- sim_adpcr(m = 100, n = 765, times = 10000, pos_sums = FALSE, n_panels = 2)
adpcr3 <- rename_dpcr(adpcr3, exper = “Experiment3“)
six_panels_example <- bind_dpcr(adpcr1, adpcr2, adpcr3)
six_panels_example <- rename_dpcr(six_panels_example, assay = factor(rep(c("Chr4", "MYC"), 3)))
## End(Not run)

summary-methods

Methods for Function summary

Description

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

Arguments

- `object` object of class `adpcr`, `dpcr` or `qdpcr`.
- `print` if `FALSE`, no output is printed.

Details

The function prints a summary of the dPCR reaction, including k (number of positive chambers), n (total number of chambers), estimated lambda and concentration, 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


Examples

```r
# 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_dpcr(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_dpcr(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")
```
**Description**

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

**Usage**

test_counts(input, model = "ratio", conf.level = 0.95)

**Arguments**

- **input**: object of class `adpcr` or `dpcr` with "nm" type.
- **model**: may have one of following values: binomial, poisson, prop, ratio. See Details.
- **conf.level**: confidence level of the intervals and groups.

**Details**

test_counts incorporates two different approaches to models: GLM (General Linear Model) and multiple pair-wise tests. The GLM fits counts data from different digital PCR experiments using quasibinomial or quasipoisson family. Comparisons between single experiments utilize Tukey’s contrast and multiple t-tests (as provided by function glht).

In case of pair-wise tests, (rateratio.test or prop.test) are used to compare all pairs of experiments. The p-values are adjusted using the Benjamini & Hochberg method (p.adjust). Furthermore, confidence intervals are simultaneous.

**Value**

an object of class `count_test`.

**Note**

Mean number of template molecules per partition and its confidence intervals will vary depending on input.

**Author(s)**

Michal Burdukiewicz, Stefan Roediger, Piotr Sobczyk.

**References**

See Also

Functions used by `test_counts`:

- `glm`,
- `glht`,
- `cld`,
- `prop.test`,
- `rateratio.test`

GUI presenting capabilities of the test: `test_counts_gui`.

Examples

```r
# be warned, the examples of test_counts are time-consuming
## Not run:
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)
adpcr2 <- rename_dpcr(adpcr2, exper = "Experiment2")
adpcr3 <- sim_adpcr(m = 10, n = 600, times = 1000, pos_sums = FALSE, n_panels = 3)
adpcr3 <- rename_dpcr(adpcr3, exper = "Experiment3")

# compare experiments using binomial regression
two_groups_bin <- test_counts(bind_dpcr(adpcr1, adpcr2), model = "binomial")
summary(two_groups_bin)
plot(two_groups_bin)
# plot aggregated results
plot(two_groups_bin, aggregate = TRUE)
# get coefficients
coef(two_groups_bin)

# this time use Poisson regression
two_groups_pois <- test_counts(bind_dpcr(adpcr1, adpcr2), model = "poisson")
summary(two_groups_pois)
plot(two_groups_pois)

# see how test behaves when results aren't significantly different
one_group <- test_counts(bind_dpcr(adpcr1, adpcr3))
summary(one_group)
plot(one_group)

## End(Not run)
```

### test_counts_gui

**Compare digital PCR runs - interactive presentation**

**Description**

Launches graphical user interface allowing multiple comparisons of simulated digital PCR reactions.
Usage

\texttt{test\_counts\_gui()}

Warning

Any ad-blocking software may be cause of malfunctions.

Author(s)

Michal Burdukiewicz, Stefan Roediger.

See Also

\texttt{test\_counts}.

\begin{verbatim}
\textbf{test\_panel} \textit{Dispersion Test for Spatial Point Pattern in Array dPCR Based on Quadrat Counts}
\end{verbatim}

Description

Performs a test of Complete Spatial Randomness for each plate. This function is a wrapper around \texttt{quadrat.test} function working directly on the objects of \texttt{adpcr}.

Usage

\texttt{test\_panel(X, nx = 5, ny = 5, alternative = c("two.sided", "regular", "clustered"), method = c("Chisq", "MonteCarlo"), conditional = TRUE, nsim = 1999)}

Arguments

\begin{itemize}
  \item \texttt{X} Object of the \texttt{adpcr} class containing data from one or more panels.
  \item \texttt{nx} Number of quadrats in the x direction.
  \item \texttt{ny} Number of quadrats in the y direction.
  \item \texttt{alternative} character string (partially matched) specifying the alternative hypothesis.
  \item \texttt{method} character string (partially matched) specifying the test to use: either "Chisq" for the chi-squared test (the default), or "MonteCarlo" for a Monte Carlo test.
  \item \texttt{conditional} logical. Should the Monte Carlo test be conducted conditionally upon the observed number of points of the pattern? Ignored if method=“Chisq”.
  \item \texttt{nsim} The number of simulated samples to generate when method="MonteCarlo".
\end{itemize}
Details

Under optimal conditions, the point pattern of dPCR events (e.g., positive droplet & negative droplets) should be randomly distributed over a planar chip. This function verifies this assumption using chi-square or Monte Carlo test. Arrays with non-random patterns should be checked for integrity.

Value

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

Note

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

Author(s)

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

References

http://www.spatstat.org/

See Also

quadrat.test.

Examples

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

# test only one plate
test_panel(extract_run(many_panels, 3))

# do test_panel manually
require(spatstat)
ppp_data <- adpcr2ppp(many_panels)
lapply(ppp_data, function(single_panel) quadrat.test(single_panel))
```
Description

Detect, separate and count positive and negative peaks, as well as peak-like noise. Additionally, 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

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$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$peaks[[3]][, 1], lty = "dashed")
for(i in 1:nrow(tmp.out$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_pooled  

Compare pooled digital PCR

Description

Estimates mean number of template molecules per partition and concentration of sample from pooled replicates of experiments.

Usage

test_pooled(input, conf.level = 0.05)

Arguments

input object of class adpcr or dpcr.
conf.level confidence level of the intervals and groups.
Value
data frame with the number of rows equal to the number of experiments (not runs). The unit of
concentration is the number template molecules per nanoliter (nL).

Note
This function was implemented using the code in supplemental materials in Dorazio, 2015 (see
References).

Author(s)
Robert M. Dorazio, Margaret E. Hunter.

References
Dorazio RM, Hunter ME, Statistical Models for the Analysis and Design of Digital Polymerase

Examples
test_pooled(six_panels)

White  Digitalized Data from a Fluidigm Array

Description
These are the results data from the White data as measured by the UT digital PCR on Fluidigm
12.765 digital Array. The data were digitized from a supplementary figure "1471-2164-10-116-
S1.pdf" by White et al. (2009) BMC Genomics

Format
A dataframe with 9180 rows and 10 columns.
Position of an array in the figure 1471-2164-10-116-S1.pdf from White et al. (2009) BMC
Genomics (e.g., 11 is the image in the first column and the first row, 24 is second column and
fourth image)

Image_position is the sample (e.g., "Ace 1:100") as described by White et al. (2009) BMC
Genomics

X.1 Running index for *all* samples
Index Index within an array
Row Row within an array
Column Column within an array

Area is the area that was measured with "MicroArray Profile"
Min is the minimum intensity of an area that was measured with "MicroArray Profile"
Max is the maximum intensity of an area that was measured with "MicroArray Profile"
Mean is the mean intensity of an area that was measured with "MicroArray Profile"
Details

Setup: Experimental details were described by White et al. (2009) BMC Genomics. The digitalization of the figure was done with imageJ and the "MicroArray Profile" plugin by Bob Dougherty (rpd@optinav.com) and Wayne Rasband.

Annotation: See the White et al. (2009) BMC Genomics paper for details.

Author(s)

Stefan Roediger, Michal Burdukiewcz, White et al. (2009) BMC Genomics

Source

Data were digitalized from the supplement material (Additional file 1. dPCR analysis of mock library control.) "1471-2164-10-116-S1.pdf" by White et al. (2009) BMC Genomics

References


Examples

```r
str(White)
par(mfrow = c(3,3))

White_data <- sapply(unique(White["Image_position"]), function(i)
    White[White["Image_position"] == i, "Mean"], simplify = TRUE)

assays <- sapply(unique(White["Image_position"]), function(i)
    unique(White[White["Image_position"] == i, "Sample"], simplify = TRUE)

White_adpcr <- create_dpcr(White_data > 115, n = 765, assay = assays,
    type = "np", adpcr = TRUE)

White_k <- colSums(White_data > 115)
sapply(2:4, function(i) {
    plot_panel(extract_run(White_adpcr, i))
    
    # Create the ECDF of the image scan data to define
    # a cut-off for positive and negative partitions
    # Plot the ECDF of the image scan data an define a cut-off
    plot(ecdf(White_data[, i]), main = paste0("ECDF of Image Scan Data\n", assays[i]),
        xlab = "Grey value", ylab = "Density of Grey values")
    abline(v = 115, col = 2, cex = 2)
    text(80, 0.5, "User defined cut-off", col = 2, cex = 1.5)
```
# Plot the density of the dPCR experiment

dpcr_density(k = White_k[i], n = 765, bars = TRUE)
}

par(mfrow = c(1,1))
Index

*Topic AUC
  test_peaks, 59
*Topic Amplitude
  bioamp, 9
*Topic Bio-Rad
  bioamp, 9
*Topic Cy0
  limit_cq, 26
  qpcr_analyser, 40
*Topic PCR
  compare_dens, 12
*Topic Poisson
  qpcr2pp, 38
*Topic Process
  qpcr2pp, 38
*Topic adPCR
  create_dpcr, 14
*Topic amplification
  qpcr_analyser, 40
*Topic classes
  adpcr-class, 4
  count_test, 12
  dpcr-class, 17
  qdpcr-class, 37
  rtadpcr-class, 47
*Topic dPCR
  limit_cq, 26
  test_panel, 57
*Topic datagen
  sim_adpcr, 49
  sim_dpcr, 51
*Topic datasets
  dPCRmethy1, 20
  many_peaks, 28
  pds, 30
  pds_raw, 32
  White, 61
*Topic ddPCR
  create_dpcr, 14

*Topic density
  compare_dens, 12
*Topic digital
  compare_dens, 12
*Topic dplot
  dpcr_density, 21
  dpcr_density_table, 23
*Topic empirical
  compare_dens, 12
*Topic extract
  extract_dpcr, 24
  extract_run, 25
*Topic hplot
  dpcr_density, 21
  dpcr_density_gui, 22
  dpcr_density_table, 23
  dpcRReport, 20
  plot.qdpcr, 33
  plot_panel, 34
  plot_vic_fam, 36
  test_counts_gui, 56
*Topic kurtosis
  momentsMmethods, 28
*Topic manip
  adpcr2panel, 5
  adpcr2pp, 6
  binarize, 7
  bind_dpcrMmethods, 8
  calc_coordinates, 11
  extract_dpcr, 24
  extract_run, 25
  num2int, 29
  rename_dpcr, 47
*Topic mean
  momentsMmethods, 28
*Topic methods
  showMmethods, 48
  summaryMmethods, 53
*Topic moments
moments.Methods, 28

*Topic noise
  test_peaks, 59
*Topic package
  dpcR.package, 3
*Topic panel
  adpcr2ppp, 6
  extract_dpcr, 24
  extract_run, 25
*Topic pattern
  test_panel, 57
*Topic peak
  test_peaks, 59
*Topic qPCR
  limit_cq, 26
  qpcr, 38
  qpcr_analyser, 40
*Topic quadrant
  test_panel, 57
*Topic quantification
  qpcr_analyser, 40
*Topic real-time
  qpcr_analyser, 40
  rtadpcr.class, 47
*Topic skewness
  moments.Methods, 28
*Topic smooth
  test_peaks, 59
*Topic spatial
  test_panel, 57
*Topic utilities
  df2dpcr, 16
  dpcr2df.Methods, 19
  read_amp, 42
  read_BioMark, 42
  read_dpcr, 43
  read_QX100, 44
  read_QX200, 45
  read_redf, 46
  show.Methods, 48
  summary.Methods, 53
*Topic variance
  moments.Methods, 28

adpcr, 5–11, 14, 16–20, 24, 25, 34, 35, 40, 42–48, 50, 52, 53, 55, 57, 59, 60
adpcr (adpcr.class), 4
adpcr-class, 4
adpcr2panel, 5, 5, 11, 35

adpcr2ppp, 6, 58
as.integer, 29
binarize, 7
bind_dpcr, 5, 26
bind_dpcr (bind_dpcr.Methods), 8
bind_dpcr, adpcr (bind_dpcr.Methods), 8
bind_dpcr, adpcr-method
  (bind_dpcr-method), 8
bind_dpcr, dpcr (bind_dpcr-method), 8
bind_dpcr, dpcr-method
  (bind_dpcr-method), 8
bind_dpcr, list (bind_dpcr-method), 8
bind_dpcr, list-method
  (bind_dpcr-method), 8
bind_dpcr-methods, 8
binom.confint, 22
bioamp, 9
BioradCNV, 10
calc_coordinates, 11, 35
cbind, 8, 9
cld, 56
coeff, count_test-method (count_test), 12
comparedens, 12
count_test, 12, 55
count_test-class (count_test), 12
create_dpcr, 14, 17
Cy0, 40
data.frame, 16, 19
ddpcRquant, 15
df2dpcr, 15, 16, 19
do.call, 9
dpcr (dpcR.package), 3
dpcr, 4, 5, 8, 9, 12, 14, 16, 17, 19, 23–25, 28, 36, 37, 43, 46–48, 52, 53, 55, 60
dpcr (dpcr-class), 17
dpcr-class, 17
dpcr-package, 3
dpcr2df, 17
dpcr2df (dpcr2df-methods), 19
dpcr2df, adpcr-method (dpcr2df-methods), 19
dpcr2df, dpcr-method (dpcr2df-methods), 19
dpcr2df-methods, 19
dpcr_density, 21, 22, 23, 35
dpcr_density_gui, 22, 22
INDEX

dpcr_density_table, 23
dpcreport, 20
dPCRmethyl, 20
efficiency, 27, 40
Extract, 24, 25
extract_dpcr, 24, 26
extract_run, 5, 9, 24, 25, 35
family, 55
findpeaks, 59
glm, 56
graphics, 11, 35
indir, 27
limit_cq, 26
many_peaks, 28
modlist, 40, 41
moments, 12
moments (moments-methods), 28
moments, dpcr-method (moments-methods), 28
moments, matrix-method
  (moments-methods), 28
moments, numeric-method
  (moments-methods), 28
moments-methods, 28
mselect, 27
new, 15
num2int, 29
p.adjust, 55
pCfit, 26, 38, 40
pds, 30, 44
pds_raw, 32
plot, count_test, ANY-method
  (count_test), 12
plot, count_test-method (count_test), 12
plot, qdPCR, ANY-method (plot.qdPCR), 33
plot, qdPCR-method (plot.qdPCR), 33
plot, count_test (count_test), 12
plot, qdPCR, 33, 38
plot_panel, 5, 11, 34, 40
plot_vic_fam, 36
ppp, 6, 7
ppp.object, 6, 7
prop.test, 55, 56
qdPCR, 18, 33, 34, 39, 53
qdPCR (qdPCR-class), 37
qdPCR-class, 37
qpcR.news, 4
qpcr2pp, 38
qPCR_analyser, 40
qPCR_analyser, adPCR-method
  (qPCR_analyser), 40
qPCR_analyser, data.frame-method
  (qPCR_analyser), 40
qPCR_analyser, modlist-method
  (qPCR_analyser), 40
qPCR_analyser-methods (qPCR_analyser), 40
quadrat.test, 57, 58
rateratio.test, 55, 56
rbind, 8
read_amp, 42
read_BioMark, 42, 44
read_dPCR, 43, 44, 45
read_QX100, 44, 44
read_QX200, 44, 45
read_redf, 44, 46
rename_dPCR, 47
rtadPCR, 5, 18
rtadPCR (rtadPCR-class), 47
rtadPCR-class, 47
show, 48
show (show-methods), 48
show, count_test-method (count_test), 12
show, dPCR-method (show-methods), 48
show, qdPCR-method (qdPCR-class), 37
show-methods, 48
show.qdPCR (qdPCR-class), 37
sim_adPCR, 5, 49, 52
sim_dPCR, 50, 51
six_panels, 52
summary, 53
summary (summary-methods), 53
summary, adPCR-method (summary-methods), 53
summary, count_test-method (count_test), 12
summary, dPCR-method (summary-methods), 53
summary, qPCR-method (qPCR-class), 37
summary-methods, 53
summary.adPCR (summary-methods), 53
summary.der, 27
summary.dPCR (summary-methods), 53
summary.qPCR (qPCR-class), 37
takeoff, 40
test_counts, 12, 13, 55, 57
test_counts_gui, 56, 56
test_panel, 5, 57
test_peaks, 59
test_peaks, adPCR-method (test_peaks), 59
test_peaks, numeric-method (test_peaks), 59
test_peaks-methods (test_peaks), 59
test_pooled, 60

White, 61