Package 'ddPCRclust'

November 11, 2026

Title Clustering algorithm for ddPCR data

Version 1.32.0

Description The ddPCRclust algorithm can automatically quantify the CPDs of non-orthogonal ddPCR reactions
with up to four targets. In order to determine the correct droplet count for each target, it is crucial to both identify all clusters and label them correctly based on their position. For more information on what data can be analyzed and how a template needs to be formatted, please check the vignette.

LazyData true

Depends R (>= 3.5)

Imports plotrix, clue, parallel, ggplot2, openxlsx, R.utils, flowCore, flowDensity (>= 1.13.3), SamSPECTRAL, flowPeaks

Suggests BiocStyle

Collate 'cluster_functions.R' 'functions.R' 'ddPCRclust.R'

License Artistic-2.0

URL https://github.com/bgbrink/ddPCRclust

BugReports https://github.com/bgbrink/ddPCRclust/issues

biocViews ddPCR, Clustering

RoxygenNote 6.0.1

git_url https://git.bioconductor.org/packages/ddPCRclust

git_branch RELEASE_3_23

git_last_commit ffa5b1c

git_last_commit_date 2026-04-28

Repository Bioconductor 3.23

Date/Publication 2026-11-11

Author Benedikt G. Brink [aut, cre],
Justin Meskas [ctb],
Ryan R. Brinkman [ctb]

Maintainer Benedikt G. Brink <bbrink@cebitec.uni-bielefeld.de>

Contents

calculateCPDs . . . 2
createEnsemble . . . 3
ddPCRclust . . . 4
exportPlots . . . 6
exportToCSV . . . 7
exportToExcel . . . 8
readFiles . . . 9
readTemplate . . . 10
runDensity . . . 11
runPeaks . . . 12
runSam . . . 13
shearCorrection . . . 14

Index **16**

`calculateCPDs` *Calculates the copies per droplet*

Description

This function takes the results of the clustering and calculates the actual counts per target, as well as the counts per droplet (CPD) for each marker.

Usage

```
calculateCPDs(results, template = NULL, constantControl = NULL)
```

Arguments

`results`	The result of the ddPCRclust algorithm.
`template`	The parsed dataframe containing the template.
`constantControl`	The constant refrence control, which should be present in each reaction. It is used to normalize the data.

Value

A list of lists, containing the counts for empty droplets, each marker with both total droplet count and CPD, and total number of droplets, for each element of the input list respectively.

Examples

```
# Read files
exampleFiles <- list.files(paste0(find.package('ddPCRclust'), '/extdata'), full.names = TRUE)
files <- readFiles(exampleFiles[3])
# To read all example files uncomment the following line
# files <- readFiles(exampleFiles[1:8])

# Read template
template <- readTemplate(exampleFiles[9])

# Run ddPCRclust
result <- ddPCRclust(files, template)

# Calculate the CPDs
markerCPDs <- calculateCPDs(result, template$template)
```

`createEnsemble` *Create a cluster ensemble*

Description

This function takes the three (or less) clustering approaches of the ddPCRclust package and combines them to one cluster ensemble. See cl_medoid for more information.

Usage

```
createEnsemble(dens = NULL, sam = NULL, peaks = NULL, file)
```

Arguments

`dens`	The result of the flowDensity algorithm as a CLUE partition.
`sam`	The result of the samSPECTRAL algorithm as a CLUE partition.
`peaks`	The result of the flowPeaks algorithm as a CLUE partition.
`file`	The input data. More specifically, a data frame with two dimensions, each dimension representing the intensity for one color.

Value

`data`	The original input data minus the removed events (for plotting)
`confidence`	The agreement between the different clustering results in percent. If all algorithms calculated the same result, the clustering is likely to be correct, thus the confidence is high.
`counts`	The droplet count for each cluster.

Examples

```
exampleFiles <- list.files(paste0(find.package('ddPCRclust'), '/extdata'), full.names = TRUE)
file <- read.csv(exampleFiles[3])
densResult <- runDensity(file = file, numOfMarkers = 4)
samResult <- runSam(file = file, numOfMarkers = 4)
peaksResult <- runPeaks(file = file, numOfMarkers = 4)

superResult <- createEnsemble(densResult, samResult, peaksResult, file)
```

`ddPCRclust` *ddPCRclust A package for automated quantification of multiplexed ddPCR data*

Description

The ddPCRclust algorithm can automatically quantify the events of ddPCR reaction with up to four markers. In order to determine the correct droplet count for each marker, it is crucial to both identify all clusters and label them correctly based on their position. For more information on what data can be analyzed and how a template needs to be formatted, please check the project repository on github.

This is the main function of this package. It automatically runs the ddPCRclust algorithm on one or multiple csv files containing the raw data from a ddPCR run with up to 4 markers.

Usage

```
ddPCRclust(files, template, numOfMarkers = 4, sensitivity = 1,
  similarityParam = 0.95, distanceParam = 0.2, fast = FALSE,
  multithread = FALSE)
```

Arguments

`files`	The input data obtained from the csv files. For more information, please see `readFiles`.
`template`	A data frame containing information about the individual ddPCR runs. An example template is provided with this package. For more information, please see `readTemplate`.
`numOfMarkers`	The number of primary clusters that are expected according the experiment set up. Can be ignored if a template is provided. Else, a vector with length equal to `length(files)` should be provided, containing the number of markers used for the respective reaction.
`sensitivity`	A number between 0.1 and 2 determining sensitivity of the initial clustering, e.g. the number of clusters. A higher value means the data is divided into more clusters, a lower value means more clusters are merged. This allows fine tuning of the algorithm for exceptionally low or high CPDs.

`similarityParam`
: If the distance of a droplet between two or more clusters is very similar, it will not be counted for either. The standard it 0.95, i.e. at least 95% similarity. A sensible value lies between 0 and 1, where 0 means none of the 'rain' droplets will be counted and 1 means all droplets will be counted.

`distanceParam`
: When assigning rain between two clusters, typically the bottom 20% are assigned to the lower cluster and the remaining 80% to the higher cluster. This parameter changes the ratio, i.e. a value of 0.1 would assign only 10% to the lower cluster.

`fast`
: Run a simpler version of the algorithm that is about 10x faster. For clean data, this might already deliver very good results. However, is is mostly intended to get a quick overview over the data.

`multithread`
: Distribute the algorithm amongst all CPU cores to speed up the computation.

Value

`results`
: The results of the ddPCRclust algorithm. It contains three fields:
`data` The original input data minus the removed events (for plotting)
`confidence` The agreement between the different clustering results in percent If all parts of the algorithm calculated the same result, the clustering is likely to be correct, thus the confidence is high
`counts` The droplet count for each cluster

Usage

The main function of the package is `ddPCRclust`. This function runs the algorithm with one or multiple files, automatically distributing them amongst all cpu cores using the parallel package (parallelization does not work on windows). Afterwards, the results can be exported in different ways, using `exportPlots`, `exportToExcel` and `exportToCSV`. Once the clustering is finished, copies per droplet (CPD) for each marker can be calculated using `calculateCPDs`.

These functions provide access to all functionalities of the ddPCRclust package. However, expert users can directly call some internal functions of the algorithm, if they find it necessary. Here is a list of all available supplemental functions:
`runDensity`
`runSam`
`runPeaks`
`createEnsemble`

Author(s)

Maintainer: Benedikt G. Brink <bbrink@cebitec.uni-bielefeld.de>

Other contributors:

- Justin Meskas <jmeskas@bccrc.ca> [contributor]
- Ryan R. Brinkman <rbrinkman@bccrc.ca> [contributor]

See Also

Useful links:

- https://github.com/bgbrink/ddPCRclust
- Report bugs at https://github.com/bgbrink/ddPCRclust/issues

Examples

```
# Read files
exampleFiles <- list.files(paste0(find.package('ddPCRclust'), '/extdata'), full.names = TRUE)
files <- readFiles(exampleFiles[3])
# To read all example files uncomment the following line
# files <- readFiles(exampleFiles[1:8])

# Read template
template <- readTemplate(exampleFiles[9])

# Run ddPCRclust
result <- ddPCRclust(files, template)

# Plot the results
library(ggplot2)
p <- ggplot(data = result$B01$data, mapping = aes(x = Ch2.Amplitude, y = Ch1.Amplitude))
p <- p + geom_point(aes(color = factor(Cluster)), size = .5, na.rm = TRUE) +
  ggtitle('B01 example')+theme_bw() + theme(legend.position='none')
p
```

`exportPlots` *Plot the algorithms results with ggplot2*

Description

A convinience function that takes the results of the ddPCRclust algorithm, plots them using the ggplot2 library and a custom colour palette and saves the plots to a folder.

Usage

```
exportPlots(data, directory, annotations, format = "png", invert = FALSE)
```

Arguments

`data`	The result of the ddPCRclust algorithm
`directory`	The parent directory where the files should saved. A new folder with the experiment name will be created (see below).
`annotations`	Some basic metadata about the ddPCR reaction. If you provided `ddPCRclust` a template, this paramater can be filled with the corresponding field in the result. Otherwise, you have to provide a character vector containing a name and the the color channels, e.g. `c(Name='ddPCR_01-04-2017', Ch1='HEX', Ch2='FAM')`

format	Which file format to use. Can be either be a device function (e.g. png), or one of 'eps', 'ps', 'tex' (pictex), 'pdf', 'jpeg', 'tiff', 'png', 'bmp', 'svg' or 'wmf' (windows only). See also ggsave
invert	Invert the axis, e.g. x = Ch2.Amplitude, y = Ch1.Amplitude

Value

None

Examples

```
# Read files
exampleFiles <- list.files(paste0(find.package('ddPCRclust'), '/extdata'), full.names = TRUE)
files <- readFiles(exampleFiles[3])
# To read all example files uncomment the following line
# files <- readFiles(exampleFiles[1:8])

# Read template
template <- readTemplate(exampleFiles[9])

# Run ddPCRclust
result <- ddPCRclust(files, template)

# Export the plots
dir.create('./Results')
exportPlots(data = result, directory = './Results/', annotations = result$annotations)
```

---

exportToCSV *Export the algorithms results to a csv file*

Description

A convinience function that takes the results of the droplClust algorithm and exports them to a csv file.

Usage

```
exportToCSV(data, directory, annotations, raw = FALSE)
```

Arguments

data	The result of the ddPCRclust algorithm
directory	The parent directory where the files should saved. A new folder with the experiment name will be created (see below).
annotations	Some basic metadata about the ddPCR reaction. If you provided ddPCRclust a template, this paramater can be filled with the corresponding field in the result. Otherwise, you have to provide a character vector containing a name and the the color channels, e.g. c(Name='ddPCR_01-04-2017', Ch1='HEX', Ch2='FAM')

raw — Boolean which determines if the annotated raw data should be exported along with the final counts. Basically, a third column will be added to the original data, which contains the cluster number to which this point was assigned to. Useful for example to visualize the clustering later on. (Warning: this can take a while!)

Value

None

Examples

```
# Read files
exampleFiles <- list.files(paste0(find.package('ddPCRclust'), '/extdata'), full.names = TRUE)
files <- readFiles(exampleFiles[3])
# To read all example files uncomment the following line
# files <- readFiles(exampleFiles[1:8])

# Read template
template <- readTemplate(exampleFiles[9])

# Run ddPCRclust
result <- ddPCRclust(files, template)

# Export the results
dir.create('./Results')
exportToCSV(data = result, directory = './Results/', annotations = result$annotations)
```

---

exportToExcel *Export the algorithms results to an Excel file*

Description

A convinience function that takes the results of the droplClust algorithm and exports them to an Excel file.

Usage

```
exportToExcel(data, directory, annotations, raw = FALSE)
```

Arguments

data — The result of the ddPCRclust algorithm

directory — The parent directory where the files should saved. A new folder with the experiment name will be created (see below).

annotations — Some basic metadata about the ddPCR reaction. If you provided ddPCRclust a template, this paramater can be filled with the corresponding field in the result. Otherwise, you have to provide a character vector containing a name and the the color channels, e.g. `c(Name='ddPCR_01-04-2017', Ch1='HEX', Ch2='FAM')`

raw — Boolean which determines if the annotated raw data should be exported along with the final counts. Basically, a third column will be added to the original data, which contains the cluster number to which this point was assigned to. Useful for example to visualize the clustering later on. (Warning: this can take a while!)

Value

None

Examples

```
# Read files
exampleFiles <- list.files(paste0(find.package('ddPCRclust'), '/extdata'), full.names = TRUE)
files <- readFiles(exampleFiles[3])
# To read all example files uncomment the following line
# files <- readFiles(exampleFiles[1:8])

# Read template
template <- readTemplate(exampleFiles[9])

# Run ddPCRclust
result <- ddPCRclust(files, template)

# Export the results
dir.create('./Results')
exportToExcel(data = result, directory = './Results/', annotations = result$annotations)
```

---

readFiles *Read the csv files from your disk*

Description

This function reads the raw csv files for ddPCRclust from disk and returns the experiment data. Please refer to the vignette for more information on how these files need to be formatted.

Usage

```
readFiles(files)
```

Arguments

files — The input file(s), specifically csv files. Each file represents a two-dimensional data frame. Each row within the data frame represents a single droplet, each column the respective intensities per colour channel.

Value

files — A data frame composed of the experiment data

ids — The file ids, e.g. A01, A02, etc.

Examples

```
# Read files
exampleFiles <- list.files(paste0(find.package('ddPCRclust'), '/extdata'), full.names = TRUE)
files <- readFiles(exampleFiles[1:8])
```

`readTemplate` *Read a template file from disk*

Description

This function reads a template file for ddPCRclust from disk and returns a run template and annotations. Please refer to the vignette for information on how this file need to be formatted.

Usage

```
readTemplate(template)
```

Arguments

`template`	A csv file containing information about the individual ddPCR runs. An example template is provided with this package. For more information, please check the vignette or the repository on github.

Value

`annotations`	The metatdata provided in the header of the template. It contains four fields: `Name` The name given to this ddPCR experiment `Ch1` Color channel 1 (usually HEX) `Ch2` Color channel 2 (usually FAM) `descriptions` Additional descriptions about this ddPCR experiment (e.g. date, exprimentor, etc.)
`template`	A parsed dataframe containing the template.

Examples

```
# Read template
exampleFiles <- list.files(paste0(find.package('ddPCRclust'), '/extdata'), full.names = TRUE)
template <- readTemplate(exampleFiles[9])
```

runDensity *Find the clusters using flowDensity*

Description

Use the local density function of the flowDensity package to find the cluster centres of the ddPCR reaction. Clusters are then labelled based on their rotated position and lastly the rain is assigned.

Usage

```
runDensity(file, sensitivity = 1, numOfMarkers, missingClusters = NULL,
  similarityParam = 0.95, distanceParam = 0.2)
```

Arguments

file	The input data. More specifically, a data frame with two dimensions, each dimension representing the intensity for one color channel.
sensitivity	A number between 0.1 and 2 determining sensitivity of the initial clustering, e.g. the number of clusters. A higher value means more clusters are being found. Standard is 1.
numOfMarkers	The number of primary clusters that are expected according the experiment set up.
missingClusters	A vector containing the number of primary clusters, which are missing in this dataset according to the template.
similarityParam	If the distance of a droplet between two or more clusters is very similar, it will not be counted for either. The standard it 0.95, i.e. at least 95% similarity. A sensible value lies between 0 and 1, where 0 means none of the 'rain' droplets will be counted and 1 means all droplets will be counted.
distanceParam	When assigning rain between two clusters, typically the bottom 20% are assigned to the lower cluster and the remaining 80% to the higher cluster. This parameter changes the ratio, i.e. a value of 0.1 would assign only 10% to the lower cluster.

Value

data	The original input data minus the removed events (for plotting)
counts	The droplet count for each cluster.
firstClusters	The position of the primary clusters.
partition	The cluster numbers as a CLUE partition (see clue package for more information).

Examples

```
# Run the flowDensity based approach
exampleFiles <- list.files(paste0(find.package('ddPCRclust'), '/extdata'), full.names = TRUE)
file <- read.csv(exampleFiles[3])
densResult <- runDensity(file = file, numOfMarkers = 4)

# Plot the results
library(ggplot2)
p <- ggplot(data = densResult$data, mapping = aes(x = Ch2.Amplitude, y = Ch1.Amplitude))
p <- p + geom_point(aes(color = factor(Cluster)), size = .5, na.rm = TRUE) +
     ggtitle('flowDensity example')+theme_bw() + theme(legend.position='none')
p
```

---

runPeaks *Find the clusters using flowPeaks*

Description

Find the rain and assign it based on the distance to vector lines connecting the cluster centres.

Usage

```
runPeaks(file, sensitivity = 1, numOfMarkers, missingClusters = NULL,
  similarityParam = 0.95, distanceParam = 0.2)
```

Arguments

file	The input data. More specifically, a data frame with two dimensions, each dimension representing the intensity for one color channel.
sensitivity	A number between 0.1 and 2 determining sensitivity of the initial clustering, e.g. the number of clusters. A higher value means more clusters are being found. Standard is 1.
numOfMarkers	The number of primary clusters that are expected according the experiment set up.
missingClusters	A vector containing the number of primary clusters, which are missing in this dataset according to the template.
similarityParam	If the distance of a droplet between two or more clusters is very similar, it will not be counted for either. The standard it 0.95, i.e. at least 95% similarity. A sensible value lies between 0 and 1, where 0 means none of the 'rain' droplets will be counted and 1 means all droplets will be counted.
distanceParam	When assigning rain between two clusters, typically the bottom 20% are assigned to the lower cluster and the remaining 80% to the higher cluster. This parameter changes the ratio, i.e. a value of 0.1 would assign only 10% to the lower cluster.

Value

data	The original input data minus the removed events (for plotting)
counts	The droplet count for each cluster.
firstClusters	The position of the primary clusters.
partition	The cluster numbers as a CLUE partition (see clue package for more information).

Examples

```
# Run the flowPeaks based approach
exampleFiles <- list.files(paste0(find.package('ddPCRclust'), '/extdata'), full.names = TRUE)
file <- read.csv(exampleFiles[3])
peaksResult <- runPeaks(file = file, numOfMarkers = 4)

# Plot the results
library(ggplot2)
p <- ggplot(data = peaksResult$data, mapping = aes(x = Ch2.Amplitude, y = Ch1.Amplitude))
p <- p + geom_point(aes(color = factor(Cluster)), size = .5, na.rm = TRUE) +
     ggtitle('flowPeaks example')+theme_bw() + theme(legend.position='none')
p
```

---

runSam *Find the clusters using SamSPECTRAL*

Description

Find the rain and assign it based on the distance to vector lines connecting the cluster centres.

Usage

```
runSam(file, sensitivity = 1, numOfMarkers, missingClusters = NULL,
  similarityParam = 0.95, distanceParam = 0.2)
```

Arguments

file	The input data. More specifically, a data frame with two dimensions, each dimension representing the intensity for one color channel.
sensitivity	A number between 0.1 and 2 determining sensitivity of the initial clustering, e.g. the number of clusters. A higher value means more clusters are being found. Standard is 1.
numOfMarkers	The number of primary clusters that are expected according the experiment set up.
missingClusters	A vector containing the number of primary clusters, which are missing in this dataset according to the template.

`similarityParam`
: If the distance of a droplet between two or more clusters is very similar, it will not be counted for either. The standard it 0.95, i.e. at least 95% similarity. A sensible value lies between 0 and 1, where 0 means none of the 'rain' droplets will be counted and 1 means all droplets will be counted.

`distanceParam`
: When assigning rain between two clusters, typically the bottom 20% are assigned to the lower cluster and the remaining 80% to the higher cluster. This parameter changes the ratio, i.e. a value of 0.1 would assign only 10% to the lower cluster.

Value

`data`
: The original input data minus the removed events (for plotting)

`counts`
: The droplet count for each cluster.

`firstClusters`
: The position of the primary clusters.

`partition`
: The cluster numbers as a CLUE partition (see clue package for more information).

Examples

```
# Run the SamSPECTRAL based approach
exampleFiles <- list.files(paste0(find.package('ddPCRclust'), '/extdata'), full.names = TRUE)
file <- read.csv(exampleFiles[3])
samResult <- runSam(file = file, numOfMarkers = 4)

# Plot the results
library(ggplot2)
p <- ggplot(data = samResult$data, mapping = aes(x = Ch2.Amplitude, y = Ch1.Amplitude))
p <- p + geom_point(aes(color = factor(Cluster)), size = .5, na.rm = TRUE) +
     ggtitle('SamSPECTRAL example')+theme_bw() + theme(legend.position='none')
p
```

---

shearCorrection *Correct for DNA shearing*

Description

Longer DNA templates produce a lower droplet count due to DNA shearing. This function normalizes the ddPCRclust result based on a stable marker of different lengths to negate the effect of differences in the lengths of the actual markers of interest. (Work in progress)

Usage

```
shearCorrection(counts, lengthControl, stableControl)
```

Arguments

`counts`	The counts per marker as provided by calculateCPDs.
`lengthControl`	The name of the length Control. If the template name is for example CPT2, the name in the template should be CPT2-125, where 125 represents the number of basepairs.
`stableControl`	The name of the stable Control used as a reference for this experiment.

Value

A linear regression model fitting the length vs ln(ratio) (see lm for details on linear regression).

Index

calculateCPDs, 2, *5*, *15*
cl_medoid, *3*
createEnsemble, 3, *5*

ddPCRclust, 4, *5–8*
ddPCRclust-package *(*ddPCRclust*)*, 4

exportPlots, *5*, 6
exportToCSV, *5*, 7
exportToExcel, *5*, 8

ggsave, *7*

lm, *15*

parallel, *5*

readFiles, *4*, 9
readTemplate, *4*, 10
runDensity, *5*, 11
runPeaks, *5*, 12
runSam, *5*, 13

shearCorrection, 14