Package 'twoddpcr'

November 12, 2026

Title Classify 2-d Droplet Digital PCR (ddPCR) data and quantify the number of starting molecules

Version 1.36.0

Author Anthony Chiu [aut, cre]

Maintainer Anthony Chiu <anthony@achiu.me>

URL http://github.com/CRUKMI-ComputationalBiology/twoddpcr/

BugReports http://github.com/CRUKMI-ComputationalBiology/twoddpcr/issues/

Description The twoddpcr package takes Droplet Digital PCR (ddPCR) droplet amplitude data from Bio-Rad's QuantaSoft and can classify the droplets. A summary of the positive/negative droplet counts can be generated, which can then be used to estimate the number of molecules using the Poisson distribution. This is the first open source package that facilitates the automatic classification of general two channel ddPCR data. Previous work includes 'definetherain' (Jones et al., 2014) and 'ddpcRquant' (Trypsteen et al., 2015) which both handle one channel ddPCR experiments only. The 'ddpcr' package available on CRAN (Attali et al., 2016) supports automatic gating of a specific class of two channel ddPCR experiments only.

Depends R (>= 3.4)

Imports class, ggplot2, hexbin, methods, scales, shiny, stats, utils, RColorBrewer, S4Vectors

License GPL-3

LazyData true

RoxygenNote 7.1.1

Collate 'KRAScounts.R' 'KRASdata.R' 'global.R' 'ddpcrWell.R' 'ddpcrPlate.R' 'clusterCentres.R' 'clusterStats.R' 'themes.R' 'dropletPlot.R' 'exportTable.R' 'facetPlot.R' 'flatPlot.R' 'ggplot.R' 'gridClassify.R' 'heatPlot.R' 'isTwoDimDataFrame.R' 'kmeansClassify.R' 'knnClassify.R' 'mahRain.R' 'normalise.R' 'numDroplets.R' 'parseCounts.R' 'plateSummary.R' 'readCSVDataFrame.R' 'relabelClasses.R' 'removeDropletClasses.R' 'sdRain.R' 'shinyVisApp.R'

'shinyVisGlobal.R' 'shinyVisServer.R' 'shinyVisUI.R'
'summaries.R' 'twoddpcr.R' 'wellNameSort.R'

Suggests devtools, knitr, reshape2, rmarkdown, testthat, BiocStyle

VignetteBuilder knitr

biocViews ddPCR, Software, Classification

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

git_branch RELEASE_3_23

git_last_commit 4723b57

git_last_commit_date 2026-04-28

Repository Bioconductor 3.23

Date/Publication 2026-11-11

Contents

twoddpcr-package . . . 4
.classifyDfOnChannel . . . 4
.classifyOnChannel . . . 5
.classwiseMahalanobisRain . . . 6
.cov . . . 7
.dependentCols . . . 7
.essentialDependentCols . . . 8
.extractWellNames . . . 8
.getAllSummary . . . 9
.getChannelCentres . . . 9
.getClassificationData . . . 10
.getMutCopies . . . 11
.getWellNames . . . 11
.getWtCopies . . . 12
.isTwoDimDataFrame . . . 12
.isWideForm . . . 13
.mahDist . . . 13
.matrixInverse . . . 14
.numberOfWells . . . 14
.renormaliseByChannel . . . 15
.renormaliseWell . . . 16
.roundIt . . . 16
.slice . . . 17
.totalCopies . . . 17
amplitudes . . . 18
basicsSummary . . . 19
castSummary . . . 20
classCov . . . 21
classMeans . . . 22
classStats . . . 23
clusterCentres . . . 24

copiesSummary 25
ddpcr 26
ddpcrPlate-class 26
ddpcrWell-class 27
drawBlank 29
dropletPlot 29
elementType,SimpleList-method 32
exportTable 33
extractPlateName 36
extractWellNames 37
facetPlot 38
flatPlot 40
fullCopiesSummary 42
fullCountsSummary 43
getCutOff 44
ggplot.well 45
gridClassify 47
heatPlot 51
isEmpty,ddpcrWell-method 53
kmeansClassify 54
knnClassify 56
KRAScounts 58
KRASdata 59
mahalanobisRain 60
mutantCopiesSummary 61
numDroplets 62
numericInputRow 63
parseClusterCounts 64
plateClassification 65
plateClassificationMethod 67
plateSummary 69
positiveCounts 70
readCSVDataFrame 71
relabelClasses 72
removeDropletClasses 73
renormalisePlate 75
sdRain 76
setChannelNames 78
setDropletVolume 79
shinyVis 80
shinyVisApp 80
shinyVisServer 81
shinyVisUI 81
sortDataFrame 82
sortWells 82
textInputRow 83
thresholdClassify 84
wellClassification 86

wellClassificationMethod 87
whiteTheme 89
wildTypeCopiesSummary 90

Index **91**

`twoddpcr-package` *Classifying and summarising 2-d droplet digitial PCR (ddPCR) data.*

Description

The `twoddpcr` package takes droplet amplitude data from Bio-Rad's QuantaSoft and can classify the droplets. A summary of the positive/negative droplet counts can be generated, which can then be used to estimate the number of molecules using the Poisson distribution.

Author(s)

Anthony Chiu, <anthony.chiu@cruk.manchester.ac.uk>

See Also

Useful links:

- http://github.com/CRUKMI-ComputationalBiology/twoddpcr/
- Report bugs at http://github.com/CRUKMI-ComputationalBiology/twoddpcr/issues/

`.classifyDfOnChannel` *K-means classify a data frame where the droplets are negative in the same channels only.*

Description

K-means classify a data frame where the droplets are negative in the same channels only.

Usage

```
.classifyDfOnChannel(
  df,
  channel,
  centres = NULL,
  minSeparation = 2000,
  fullTable = TRUE
)
```

Arguments

df	A data frame corresponding to a well with droplets corresponding only to "NN" and "NP" or "NN" and "PN".
channel	The channel on which to classify (1 or 2).
centres	A data frame of centres. The data frame should have columns Ch1.Amplitude and Ch2.Amplitude and row names corresponding the cluster label, e.g. "NN", "NP", "PN" or "PP".
minSeparation	The minimum distance required between two cluster centres in order for us to assume that k-means found two distinct clusters. Defaults to 2000.
fullTable	Whether to return a full table or just a vector. Defaults to 'TRUE'

Value

A classification for df.

Author(s)

Anthony Chiu, <anthony.chiu@cruk.manchester.ac.uk>

.classifyOnChannel *K-means classify a list of data frames individually, where each data frame comprises droplets that are negative in the same channels only.*

Description

K-means classify a list of data frames individually, where each data frame comprises droplets that are negative in the same channels only.

Usage

```
.classifyOnChannel(
  cl,
  channel,
  centres = NULL,
  minSeparation = 2000,
  fullTable = TRUE
)
```

Arguments

cl	List of data frames, where each data frame corresponds to a well with droplets corresponding only to "NN" and "NP" or "NN" and "PN".
channel	The channel on which to classify (1 or 2).
centres	A data frame of centres. The data frame should have columns Ch1.Amplitude and Ch2.Amplitude and row names corresponding the cluster label, e.g. "NN", "NP", "PN" or "PP".

minSeparation	The minimum distance required between two cluster centres in order for us to assume that k-means found two distinct clusters. Defaults to 2000.
fullTable	Whether to return a full table or just a vector. Defaults to 'TRUE'

Value

A classification for cl.

Author(s)

Anthony Chiu, <anthony.chiu@cruk.manchester.ac.uk>

.classwiseMahalanobisRain

Fuzzy clusters by bivariate normal distributions.

Description

Assume that the class cl is bivariate normally distributed. This method adds fuzziness to the class. Other classes are not changed.

Usage

```
.classwiseMahalanobisRain(droplets, cl, maxDistance = 30, classCol = "class")
```

Arguments

droplets	A data frame of droplets with Ch1.Amplitude and Ch2.Amplitude columns, as well as a class column (see the parameter classCol).
cl	The class to focus on. This should be either "NN", "NP", "PN" or "PP".
maxDistance	An integer corresponding to the maximum Mahalanobis distance for which we will consider points to be members of the class, i.e. this is the level outside of which we consider droplets to be too far from the cluster.
classCol	The column (name or number) from droplets representing the class.

Value

A factor corresponding to the class column with Rain entries added for the class cl.

Author(s)

Anthony Chiu, <anthony.chiu@cruk.manchester.ac.uk>

.cov *Get the covariance of a cluster.*

Description

Get the covariance of a single cluster.

Usage

```
.cov(cl, droplets, classCol)
```

Arguments

cl	The cluster of which to find the covariance.
droplets	A data frame of droplets with Ch1.Amplitude and Ch2.Amplitude columns, as well as a class column (see the parameter classCol).
classCol	The column (name or number) from droplets representing the class.

Value

The covariance matrix of the chosen cluster. If not defined, return NULL.

Author(s)

Anthony Chiu, <anthony.chiu@cruk.manchester.ac.uk>

.dependentCols *Get a vector of all dependent columns.*

Description

Get a list of the columns in the ddPCR data that depend on the target (e.g. mutant or wild type). This includes columns that cannot be calculated from the very basic droplet counts.

Usage

```
.dependentCols(prefix = "")
```

Arguments

prefix	A string to prepend to each of the column names. Defaults to "" (the empty string).

Value

A vector of column names with an optional prefix.

Author(s)

Anthony Chiu, <anthony.chiu@cruk.manchester.ac.uk>

.essentialDependentCols *Get a vector of essential dependent columns.*

Description

Get a list of the essential columns in the ddPCR data that depend on the target (e.g. mutant or wild type).

Usage

```
.essentialDependentCols(prefix = "")
```

Arguments

prefix	A string to prepend to each of the essential column names. Defaults to "" (the empty string).

Value

A vector of essential column names with an optional prefix.

Author(s)

Anthony Chiu, <anthony.chiu@cruk.manchester.ac.uk>

.extractWellNames *Retrieve the well names to use from a given list.*

Description

We simply retrieve the names from a given list, but also perform some checks to make sure that the names are consistent with a given ddpcrPlate object.

Usage

```
.extractWellNames(theObject, aList)
```

Arguments

theObject	A ddpcrPlate object.
aList	A list from which we wish to extract well names.

Value

The names of `aList`.

Author(s)

Anthony Chiu, <anthony.chiu@cruk.manchester.ac.uk>

.getAllSummary *Get a summary of the number of molecules in 20ul.*

Description

Returns a data frame with a summary of the number of mutant and wild type copies per 20ul.

Usage

```
.getAllSummary(df)
```

Arguments

`df`	A data frame generated by `fullCountsSummary`.

Value

A data frame with mutant and wild type copies per 20ul, the total number of copies per 20ul, and a flag indicating if there are more than 2 mutant copies per 20ul.

Author(s)

Anthony Chiu, <anthony.chiu@cruk.manchester.ac.uk>

.getChannelCentres *Find the centres of each of the wells in a given channel.*

Description

The plate should have been classified by k-means clustering.

Usage

```
.getChannelCentres(plate, cMethod, channel, minSeparation = 2000)
```

Arguments

plate	A ddpcrPlate object from which to extract the centres.
cMethod	The classification method to use (in the form of a character string).
channel	An integer 1 or 2 corresponding to the channel of interest.
minSeparation	The minimum distance required between two cluster centres in order for us to assume that k-means found two distinct clusters. Defaults to 2000.

Value

A list of data frames, where each data frame has information about the centres in the channel of interest.

Author(s)

Anthony Chiu, <anthony.chiu@cruk.manchester.ac.uk>

.getClassificationData

Extract a classification from a data frame.

Description

Check that non-Ch*Amplitude columns are in a known classification format and coerce it to a factor.

Usage

```
.getClassificationData(colName, well)
```

Arguments

colName	The name of the column to focus on.
well	The data frame from which to extract the classifications.

Value

A factor with levels in ddpcr$classesRain.

Author(s)

Anthony Chiu, <anthony.chiu@cruk.manchester.ac.uk>

.getMutCopies *Get the mutant copies per 20ul of a data frame.*

Description

Returns a data frame with the number of mutant copies per 20ul.

Usage

```
.getMutCopies(df)
```

Arguments

df	A data frame generated by fullCountsSummary.

Value

A data frame with the basic columns and mutant copies per 20ul.

Author(s)

Anthony Chiu, <anthony.chiu@cruk.manchester.ac.uk>

.getWellNames *Extract the well names from a data frame.*

Description

Get the well names from a data frame, checking if there is a Well column or by using the row numbering.

Usage

```
.getWellNames(df)
```

Arguments

df	A data frame.

Value

A factor of row names.

Author(s)

Anthony Chiu, <anthony.chiu@cruk.manchester.ac.uk>

.getWtCopies — *Get the wild type copies per 20ul of a data frame.*

Description

Returns a data frame with the number of wild type copies per 20ul.

Usage

```
.getWtCopies(df)
```

Arguments

df	A data frame generated by fullCountsSummary.

Value

A data frame with the basic columns and wild type copies per 20ul.

Author(s)

Anthony Chiu, <anthony.chiu@cruk.manchester.ac.uk>

.isTwoDimDataFrame — *Checks whether an object is a data frame with two leading double columns.*

Description

Checks whether an object is a data frame with two leading double columns.

Usage

```
.isTwoDimDataFrame(df)
```

Arguments

df	A data frame.

Value

A logical value: whether the data frame meets the criteria.

.isWideForm *Checks a data frame is a wide-form table.*

Description

Our preferred data frame format is to have things in a wide-form data frame, i.e. to have channel 1 and channel 2 data both in the same row.

Usage

```
.isWideForm(df, ch1Label = "Mt", ch2Label = "Wt")
```

Arguments

df	A data frame.
ch1Label	The prefix to use for the channel 1 target. Defaults to "Mt".
ch2Label	The prefix to use for the channel 2 target. Defaults to "Wt".

Value

TRUE if df is considered to be of the correct format and FALSE otherwise.

Author(s)

Anthony Chiu, <anthony.chiu@cruk.manchester.ac.uk>

.mahDist *Mahalanobis distance of droplets from a distribution.*

Description

Find the Mahalanobis distance of all droplets from a given distribution.

Usage

```
.mahDist(droplets, clusStats)
```

Arguments

droplets	A data frame of droplets with Ch1.Amplitude and Ch2.Amplitude columns.
clusStats	A list of statistics for a cluster generated by classStats. This should have a mean, cov and cov.inv values.

Value

A vector of numbers, where each one corresponds to the distance of that droplet from the given distribution.

Author(s)

Anthony Chiu, <anthony.chiu@cruk.manchester.ac.uk>

`.matrixInverse` *Get the inverse of a matrix*

Description

Get the inverse of a matrix

Usage

```
.matrixInverse(s)
```

Arguments

`s`	A 2x2 matrix.

Value

The inverse of 's', or NULL if the matrix is singular

Given a matrix, compute the inverse or return `NULL` if it is singular.

`.numberOfWells` *Find the number of wells in the data frame.*

Description

Auxiliary function to find the number of wells that appear in the data frame, each with the same frequency (almost certainly once or twice). If some wells appear more than others, the function stops because the data is in an unknown format.

Usage

```
.numberOfWells(df)
```

Arguments

`df`	A data frame of droplet counts with a `Well` column.

Details

Bio-Rad's QuantaSoft produces data in long form tables, i.e. for each well there is a row for each target such as mutant and wild type. In this package, we prefer to work with wide form tables that combine the mutant and wild type parts.

Value

An integer giving the number of wells.

Author(s)

Anthony Chiu, <anthony.chiu@cruk.manchester.ac.uk>

.renormaliseByChannel *Normalise a well on one channel only and then transform it back to the original (combined) scale.*

Description

Normalise a well on one channel only and then transform it back to the original (combined) scale.

Usage

```
.renormaliseByChannel(wellDf, combinedCentres, wellCentres, channel)
```

Arguments

wellDf	A data frame of the well's droplet amplitudes.
combinedCentres	A data frame of the combined (average) centres of the non-normalised wells.
wellCentres	A data frame of centres corresponding to the given channel.
channel	An integer 1 or 2 corresponding to the channel that we are interested in.

Value

A data frame with the rescaled amplitudes in the chosen channel.

Author(s)

Anthony Chiu, <anthony.chiu@cruk.manchester.ac.uk>

.renormaliseWell — *Normalise a well in both channels and then transform it back to the original (combined) scale.*

Description

Normalise a well in both channels and then transform it back to the original (combined) scale.

Usage

```
.renormaliseWell(allDf, well, combinedCentres, indivCentres1, indivCentres2)
```

Arguments

allDf	A list of data frames, where each one corresponds to a well's droplet amplitudes.
well	The name or number of the well to normalise.
combinedCentres	A data frame of the combined (average) centres of the non-normalised wells.
indivCentres1	A data frame of centres corresponding to channel 1.
indivCentres2	A data frame of centres corresponding to channel 2.

Value

A list of data frames with the rescaled amplitudes in both channels.

Author(s)

Anthony Chiu, <anthony.chiu@cruk.manchester.ac.uk>

.roundIt — *Round to at least n decimal places.*

Description

For a data frame, if an extry is < 1, then round it to n significant figures. If it is >= 1, then round it to n decimal places.

Usage

```
.roundIt(df, n = 3)
```

Arguments

df	A data frame.
n	How many decimal places/significant figures to round to.

Value

The data frame x with rounded entries.

.slice *Splits a long vector and according to a vector of sizes.*

Description

Takes one long vector/factor and splits it into a list of vectors, where the lengths are given by a vector of sizes. This may be the same as another given list of vectors. Particularly useful if we want to combine a list of data, do some analysis on the combined data, then split the analysis.

Usage

```
.slice(vec, wellSizes, wellNames)
```

Arguments

vec	The vector to split.
wellSizes	A numeric vector corresponding to sizes of the wells.
wellNames	A character vector corresponding to the names of the wells.

Value

A list of vectors split into the given lengths.

Author(s)

Anthony Chiu, <anthony.chiu@cruk.manchester.ac.uk>

.totalCopies *Get the total number of molecules in 20ul.*

Description

Get the total number of copies of mutant and wild type molecules in a 20ul well.

Usage

```
.totalCopies(df)
```

Arguments

df	An input data frame with copies per 20ul figures.

Value

A vector listing the total numbers of copies per 20ul well.

Author(s)

Anthony Chiu, <anthony.chiu@cruk.manchester.ac.uk>

`amplitudes` *Retrieve droplet amplitudes.*

Description

Retrieve the droplet amplitudes from an object.

Usage

```
amplitudes(theObject)

## S4 method for signature 'ddpcrWell'
amplitudes(theObject)

## S4 method for signature 'ddpcrPlate'
amplitudes(theObject)
```

Arguments

`theObject`	A `ddpcrWell` or `ddpcrPlate` object.

Value

If `theObject` is a `ddpcrWell` object, return a data frame of droplet amplitudes with columns "Ch1.Amplitude" and "Ch2.Amplitude". If `theObject` is a `ddpcrPlate` object, return a list of data frames.

Author(s)

Anthony Chiu, <anthony.chiu@cruk.manchester.ac.uk>

See Also

`wellClassification` for the classification of the droplets.

Examples

```
## Set a ddpcrWell object with no data.
aWell <- ddpcrWell(well=data.frame("Ch1.Amplitude"=double(),
                                   "Ch2.Amplitude"=double()))

## This can be checked to be empty.
amplitudes(aWell)

## Alternatively, load some data.
aWell <- ddpcrWell(well=KRASdata[["E03"]])

## We check again and see that it has been populated.
head(amplitudes(aWell))

# Get all of the KRASdata droplet amplitudes.
krasPlate <- ddpcrPlate(wells=KRASdata)
allDroplets <- amplitudes(krasPlate)
str(allDroplets)
```

---

basicsSummary *Get the very basic columns of a data frame.*

Description

It is useful to see the droplet counts for each class in addition to the overall droplet count at a glance.

Usage

```
basicsSummary(df)
```

Arguments

df	A data frame generated by fullCountsSummary.

Value

A data frame with the PP, PN, NP, NN and AcceptedDroplets figures.

Author(s)

Anthony Chiu, <anthony.chiu@cruk.manchester.ac.uk>

Examples

```
df <- fullCountsSummary(KRAScountsQS)
basicsSummary(df)
```

`castSummary` *Makes a long form data frame into wide form.*

Description

Returns a data frame with the dependent columns prefixed with given labels (depending on the targets). All relevant columns are retained.

Usage

```
castSummary(df, ch1Label = "Mt", ch2Label = "Wt", rows = NULL)
```

Arguments

`df`	A data frame created by calling `read.csv` on the raw ddPCR output.
`ch1Label`	The prefix to use for the channel 1 target. Defaults to "Mt".
`ch2Label`	The prefix to use for the channel 2 target. Defaults to "Wt".
`rows`	The number of rows to retain from the original data frame. If `NULL`, all of the wells are used. Defaults to `NULL`.

Value

A data frame with the target rows merged.

Author(s)

Anthony Chiu, <anthony.chiu@cruk.manchester.ac.uk>

Examples

```
## Cast output from Bio-Rad's QuantaSoft into wide form.
castSummary(KRAScountsQS)

## Only retain selected rows.
castSummary(KRAScountsQS, rows=c(1,4:6))
```

classCov *Get the covariance of each cluster.*

Description

After classifying droplets, we can compute the covariance for each class.

Usage

```
classCov(droplets, classCol = "class")
```

Arguments

droplets	A data frame of droplets with Ch1.Amplitude and Ch2.Amplitude columns, as well as a class column (see the parameter classCol).
classCol	The column (name or number) from droplets representing the class.

Value

A list of covariance matrices of each cluster.

Author(s)

Anthony Chiu, <anthony.chiu@cruk.manchester.ac.uk>

Examples

```
## Get the covariance matrix of the clusters.
aWell<- KRASdata[["E03"]]
aWell$Cluster <- relabelClasses(aWell, classCol="Cluster")
classCov(aWell, classCol="Cluster")

## We repeat the above but with a sample with no "PP" cluster.
aWell <- KRASdata[["H04"]]
aWell$Cluster <- relabelClasses(aWell, classCol="Cluster")
classCov(aWell, classCol="Cluster")
```

`classMeans` *Get the mean of each cluster.*

Description

After classifying droplets, we can compute the mean for each class.

Usage

```
classMeans(droplets, classCol = "class")
```

Arguments

`droplets`	A data frame of droplets with `Ch1.Amplitude` and `Ch2.Amplitude` columns, as well as a classification column (see the parameter `classCol`).
`classCol`	The column (name or number) from `droplets` representing the class.

Value

A list or data frame of means of each class.

Author(s)

Anthony Chiu, <anthony.chiu@cruk.manchester.ac.uk>

Examples

```
## Get the means of the clusters.
aWell <- KRASdata[["E03"]]
aWell$Cluster <- relabelClasses(aWell, classCol="Cluster")
classMeans(aWell, classCol="Cluster")

## We repeat the above but with a sample with no "PP" cluster.
aWell <- KRASdata[["H04"]]
aWell$Cluster <- relabelClasses(aWell, classCol="Cluster")
classMeans(aWell, classCol="Cluster")
```

`classStats` *Get some basic statistical properties for each class.*

Description

This function gives the mean, covariance and inverse of the covariance for each of the classes.

Usage

```
classStats(droplets, classCol = "class")
```

Arguments

`droplets`	A data frame of droplets with `Ch1.Amplitude` and `Ch2.Amplitude` columns, as well as a class column (see the parameter `classCol`).
`classCol`	The column (name or number) from `droplets` representing the class.

Value

A list (grouped by class name) of lists with keys `mean`, `cov` and `cov.inv`. If `cov` is a singular matrix, then `cov.inv` will be `NULL`.

Author(s)

Anthony Chiu, <anthony.chiu@cruk.manchester.ac.uk>

Examples

```
## Get some basic statistical properties of the clusters.
aWell <- KRASdata[["E03"]]
aWell$Cluster <- relabelClasses(aWell, classCol="Cluster")
classStats(aWell, classCol="Cluster")

## We repeat the above but with a sample with no "PP" cluster.
aWell <- KRASdata[["H04"]]
aWell$Cluster <- relabelClasses(aWell, classCol="Cluster")
classStats(aWell, classCol="Cluster")
```

clusterCentres *Retrieve the cluster centres.*

Description

clusterCentres retrieves the cluster centres for a ddpcrWell object or the centres for each well in a ddpcrPlate object.

combinedCentres retrieves the cluster centres for all of the wells together.

Usage

```
clusterCentres(theObject, cMethod)

## S4 method for signature 'ddpcrWell'
clusterCentres(theObject, cMethod)

## S4 method for signature 'ddpcrPlate'
clusterCentres(theObject, cMethod)

combinedCentres(theObject, cMethod)

## S4 method for signature 'ddpcrPlate'
combinedCentres(theObject, cMethod)
```

Arguments

theObject	A ddpcrWell or ddpcrPlate object.
cMethod	The classification method for which to obtain the centres.

Value

If a ddpcrWell object is given, clusterCentres returns the cluster centres as a data frame.

If a ddpcrPlate object is given, clusterCentres return a list of data frames, where each data frame corresponds to the cluster centres of a well.

combinedCentres returns a data frame of the centres of all of the wells combined.

Author(s)

Anthony Chiu, <anthony.chiu@cruk.manchester.ac.uk>

Examples

```
## Get the centres of a sample with 4 clusters.
aWell <- ddpcrWell(well=KRASdata[["E03"]])
clusterCentres(aWell, "Cluster")

## Get the centres of a sample with 3 clusters.
```

```
aWell <- ddpcrWell(well=KRASdata[["H04"]])
clusterCentres(aWell, "Cluster")

## Retrieve the cluster centres of each of the wells in a \code{ddpcrPlate}
## object.
krasPlate <- ddpcrPlate(wells=KRASdata)
clusterCentres(krasPlate, cMethod="Cluster")

## Retrieve the cluster centres of all wells combined.
combinedCentres(krasPlate, cMethod="Cluster")
```

`copiesSummary` *Get the total copies per 20ul of a data frame in the context of the basic counts.*

Description

Returns a data frame with the basic figures and a summary of the number of copies per 20ul.

Usage

```
copiesSummary(df)
```

Arguments

`df`	A data frame generated by `fullCountsSummary`.

Value

A data frame with the basic data columns, mutant and wild type copies per 20ul, the total number of copies per 20ul, and a flag indicating if there are more than 2 mutant copies per 20ul.

Author(s)

Anthony Chiu, <anthony.chiu@cruk.manchester.ac.uk>

Examples

```
df <- fullCountsSummary(KRAScountsQS)
copiesSummary(df)
```

ddpcr *An environment for package variables.*

Description

Stores default variables used in the package.

Usage

```
ddpcr
```

Format

An object of class `environment` of length 10.

Value

An environment for storing package variables.

ddpcrPlate-class *An S4 class for multiple wells in a ddPCR experiment.*

Description

An S4 class for multiple wells in a ddPCR experiment.

The constructor for the `ddpcrPlate` class.

Usage

```
ddpcrPlate(wells)

## S4 method for signature 'list'
ddpcrPlate(wells)

## S4 method for signature 'ddpcrPlate'
ddpcrPlate(wells)

## S4 method for signature 'character'
ddpcrPlate(wells)

## S4 method for signature 'missing'
ddpcrPlate(wells)

## S4 method for signature 'ddpcrPlate'
show(object)
```

Arguments

wells	Either: • a list of ddpcrWell objects, • a list of data frames of droplet amplitudes, or • a character vector corresponding to a file path(s) containing CSV files of raw droplet amplitude data.
object	Any R object

Value

A ddpcrPlate object with the given wells.

Author(s)

Anthony Chiu, <anthony.chiu@cruk.manchester.ac.uk>

Anthony Chiu, <anthony.chiu@cruk.manchester.ac.uk>

Examples

```
## A \code{ddpcrPlate} object can be created from data from a list of data
## frames.
ddpcrPlate(KRASdata)

## A directory (or individual files) of droplet amplitude CSVs can also be
## loaded.
moreAmpsDir <- system.file("extdata", "more-amplitudes", package="twoddpcr")
ddpcrPlate(moreAmpsDir)
```

---

ddpcrWell-class *An S4 class for the classification of a single well in a ddPCR experiment.*

Description

An S4 class for the classification of a single well in a ddPCR experiment.

The constructor for the ddpcrWell class.

Usage

```
ddpcrWell(well)

## S4 method for signature 'data.frame'
ddpcrWell(well)

## S4 method for signature 'character'
```

```
ddpcrWell(well)

## S4 method for signature 'missing'
ddpcrWell(well)

## S4 method for signature 'ddpcrWell'
ddpcrWell(well)

## S4 method for signature 'ddpcrWell'
show(object)
```

Arguments

well	A well with columns Ch1.Amplitude and Ch2.Amplitude and optional classification columns. This can be in the form of a data frame or the path to a droplet amplitude CSV file.
object	Any R object

Value

A ddpcrWell object with the given droplets in the well.

Slots

dropletAmplitudes A data frame with columns Ch1.Amplitude and Ch2.Amplitude corresponding to all the droplets in the ddPCR well.

classification A vector of factors, where the levels are given by ddpcr$classesRain.

Author(s)

Anthony Chiu, <anthony.chiu@cruk.manchester.ac.uk>

Anthony Chiu, <anthony.chiu@cruk.manchester.ac.uk>

Examples

```
## A \code{ddpcrWell} object can be created from data from a list of data
## frames.
ddpcrWell(KRASdata[[1]])

## An CSV file of droplet amplitudes can also be loaded.
ampFile <- system.file("extdata/amplitudes/sample_B03_Amplitude.csv",
                       package="twoddpcr")
ddpcrWell(ampFile)
```

drawBlank — *Plot nothing.*

Description

We may sometimes request that the `dropletPlot` method takes an empty data frame as its argument, e.g. an empty CSV file is loaded. This normally presents an error, which is not the desired output. This function plots nothing to show that there was no data to plot.

Usage

```
drawBlank()
```

Value

A blank `ggplot` object.

See Also

`ggplot.well` and `ggplot.plate` are wrappers for plotting `ddpcrWell` and `ddpcrPlate` objects.

If there is at least one droplet, the `dropletPlot` method plots droplet amplitude classifications.

dropletPlot — *Plot a droplet classification with a colour-blind palette, optional cluster centres and fixed axes.*

Description

Plot an object comprising droplet amplitudes and their classification. If specified, centres of clusters can be marked, e.g. k-means clustering can take a set of centres as the initial centres of the algorithm, and the algorithm also outputs the final cluster centres. Limits to the axes can also be set for ease of comparison and consistency.

If a `ggplot` object is given as a parameter, this method will simply plot it with the pretty colours, centres and restrictions on the axes.

If a `data.frame` is given as a parameter, it should correspond to droplets with their classification.

If a `ddpcrWell` object is given as a parameter, plot the droplets in the well with its classification.

If a `ddpcrPlate` object is given as a parameter, plot the droplets from all wells with their classifications.

Usage

```
dropletPlot(
  droplets,
  ch1Label = "Ch1 Amplitude",
  ch2Label = "Ch2 Amplitude",
  ...
)

## S4 method for signature 'data.frame'
dropletPlot(
  droplets,
  ch1Label = "Ch1 Amplitude",
  ch2Label = "Ch2 Amplitude",
  cMethod = "None",
 mapping = aes_string(x = "Ch2.Amplitude", y = "Ch1.Amplitude", colour = cMethod),
  finalCentres = NULL,
  initialCentres = NULL,
  selectedCentre = NULL,
  pointSize = 1,
  plotLimits = list(x = c(1000, 9000), y = c(3000, 13500)),
  legendLabels = ddpcr$classesRain
)

## S4 method for signature 'ddpcrWell'
dropletPlot(
  droplets,
  ch1Label = "Ch1 Amplitude",
  ch2Label = "Ch2 Amplitude",
  cMethod = "None",
 mapping = aes_string(x = "Ch2.Amplitude", y = "Ch1.Amplitude", colour = cMethod),
  finalCentres = NULL,
  initialCentres = NULL,
  selectedCentre = NULL,
  pointSize = 1,
  plotLimits = list(x = c(1000, 9000), y = c(3000, 13500)),
  legendLabels = ddpcr$classesRain
)

## S4 method for signature 'ddpcrPlate'
dropletPlot(
  droplets,
  ch1Label = "Ch1 Amplitude",
  ch2Label = "Ch2 Amplitude",
  cMethod = "None",
 mapping = aes_string(x = "Ch2.Amplitude", y = "Ch1.Amplitude", colour = cMethod),
  finalCentres = NULL,
  initialCentres = NULL,
  selectedCentre = NULL,
```

```
  pointSize = 1,
  plotLimits = list(x = c(1000, 9000), y = c(3000, 13500)),
  legendLabels = ddpcr$classesRain
)
```

Arguments

droplets	An object corresponding to droplet amplitudes and their classifications. This can be in the form of: • A data frame with columns Ch1.Amplitude, Ch2.Amplitude and a classification column (see the parameter cMethod). • A ddpcrWell object. • A ddpcrPlate object. • A ggplot (gg) object. For example, this could be the output of ggplot.well or ggplot.plate. We should not need to use this unless we are writing new methods to plot new data types.
ch1Label	The label for the channel 1 target. Defaults to "Ch1 Amplitude".
ch2Label	The label for the channel 2 target. Defaults to "Ch2 Amplitude".
...	Other plotting parameters that depend on the object type of droplets.
cMethod	This should be the name or column number of droplets corresponding to the classification. This column should only have entries in "NN", "PN", "NP, "PP", "Rain" and "N/A". If "None", plots the droplets with all of them classified as N/A. Defaults to "None".
mapping	A list of aesthetic mappings to use for the plot. Defaults to ggplot2::aes_string(x="Ch2.Amplitude", y="Ch1.Amplitude",colours=cMethod). Not used if droplets is a ggplot object.
finalCentres	A data frmae of final centres to plot (e.g. those returned by the k-means or c-means algorithms). If NULL, nothing is plotted. Defaults to NULL.
initialCentres	A data frame of initial centres to plot (e.g. initial cluster centres used in the k-means). If NULL, nothing is plotted. Defaults to NULL. This parameter is useful for illustrative reasons.
selectedCentre	An initial centre to highlight. This should be either "NN", "NP", "PN" or "PP". If NULL, nothing is highlighted. Defaults to NULL. This parameter is useful for illustrative reasons.
pointSize	The size to draw each droplet. Defaults to 1.
plotLimits	A list of 2-element vectors with names x and y. These are used to fix the x and y limits of the plot, which is especially useful for comparing plots. Defaults to list(x=c(1000,9000), y=c(3000, 13500)).
legendLabels	The character vector corresponding to the labels for the legend. The elements of the vector should correspond to the NN, NP, PN, PP, Rain and N/A classes, respectively. Defaults to ddpcr$classesRain.

Value

A ggplot object with all of the given information above.

Author(s)

Anthony Chiu, <anthony.chiu@cruk.manchester.ac.uk>

Examples

```
## Get a data frame and relabel the "Cluster" column to the right form.
aWell <- KRASdata[["E03"]]
aWell$Cluster <- relabelClasses(aWell, classCol="Cluster")

## Plot the data frame.
dropletPlot(aWell, cMethod="Cluster")

## Plot a ddpcrWell object.
aWell <- ddpcrWell(well=KRASdata[["E03"]])
dropletPlot(aWell, cMethod="Cluster")

## Plot a ddpcrPlate object.
krasPlate <- ddpcrPlate(wells=KRASdata[c("E03", "H03", "C04", "F04")])
dropletPlot(krasPlate, cMethod="Cluster")

## Use K-means clustering to classify a single sample. Then plot the
## classification and final cluster centres.
aWell <- kmeansClassify(aWell)
centres <- clusterCentres(aWell, cMethod="kmeans")
dropletPlot(aWell, cMethod="kmeans", finalCentres=centres)
```

---

elementType,SimpleList-method

Check the types of the elements in a `SimpleList`.

Description

Check the types of the elements in a `SimpleList`.

Usage

```
## S4 method for signature 'SimpleList'
elementType(x)
```

Arguments

x	An object.

Value

The types of objects in the `SimpleList` object.

exportTable *Exports an object to file.*

Description

If given a data frame, exportTable exports the whole data frame to file. This could be a data frame of any form. A few options are available that can be used to determine the format of the file that is exported, e.g. using a heading for the row names 'column', or omitting row names altogether.

If a ddpcrWell is given, exportTable exports to a single file with specified/all classification methods.

If a ddpcrPlate is given, exportTable exports to a directory in the given location, where one file is created for each of the wells. If it does not exist, the directory location will be created as long as all other parent directories exist.

exportZip takes a ddpcrPlate object and exports it as a zip file.

Usage

```
exportTable(theObject, location, delim = ",", ...)

## S4 method for signature 'data.frame'
exportTable(
  theObject,
  location,
  delim = ",",
  leadingColName = NULL,
  row.names = TRUE
)

## S4 method for signature 'ddpcrWell'
exportTable(theObject, location, delim = ",", cMethod = NULL)

## S4 method for signature 'ddpcrPlate'
exportTable(
  theObject,
  location,
  delim = ",",
  cMethod = NULL,
  prefix = "",
  suffix = "_Amplitude.csv"
)

exportZip(
  theObject,
  location,
  delim = ",",
  cMethod = NULL,
```

```
  prefix = "",
  suffix = "_Amplitude.csv"
)

## S4 method for signature 'ddpcrPlate'
exportZip(
  theObject,
  location,
  delim = ",",
  cMethod = NULL,
  prefix = "",
  suffix = "_Amplitude.csv"
)
```

Arguments

theObject	The dataframe to export.
location	The location to export to. This should be a filename if we are using exportZip, or we are using exportTable and theObject is a data frame or ddpcrWell object. If theObject is a ddpcrPlate object, this should be a directory.
delim	The character to use as a field separator. Defaults to ",", i.e. export a CSV.
...	Other options depending on the type of theObject.
leadingColName	The name of the leading column, i.e. the 'row names' of the dataframe. This could be a patient identifier or the well used in the ddPCR experiment. If NULL, the exported heading will be an empty string. Defaults to NULL.
row.names	If NULL, exports a column corresponding to the row names; if FALSE, no such column is included. If 'leadingColName' is not FALSE, row.names is assumed to be FALSE. Defaults to TRUE.
cMethod	The name or column number of the classification methods in a ddpcrWell or ddpcrPlate object to export to file. If NULL, all of the classification methods are exported. Defaults to NULL.
prefix	For ddpcrPlate objects, this is the prefix to prepend to the output filenames.
suffix	For ddpcrPlate objects, this is the suffix to append to the output filenames. This is typically the filename extension, e.g. ".csv" or ".txt". Defaults to ".csv".

Details

Note that filenames of the form Anything_A01_Amplitude.csv can be read by readCSVDataFrame so that the well name can be extracted successfully (in this case A01). Where it is used, see the default value of the parameter suffix.

Value

Exports a file.

Author(s)

Anthony Chiu, <anthony.chiu@cruk.manchester.ac.uk>

Examples

```
## Output to a temporary directory.
tmpOut <- file.path(normalizePath(tempdir()))

## Read some counts data and generate a summary data frame.
df <- fullCountsSummary(KRAScountsQS)
summaryDf <- fullCopiesSummary(df)

## Write the summary to a CSV file.
exportTable(summaryDf, file.path(tmpOut, "summary-table.csv"))

## Write the summary to a tab-separated text file.
exportTable(summaryDf, file.path(tmpOut, "summary-table.txt"), delim="\t")

## Write the summary to a CSV file with leading column labelled "Patient".
exportTable(summaryDf, file.path(tmpOut, "summary-table.csv"),
            leadingColName="Patient")

## Read a droplet amplitude CSV file to a ddpcrWell object.
ampFile <- system.file("extdata", "amplitudes", "sample_B03_Amplitude.csv",
                       package="twoddpcr")
aWell <- ddpcrWell(well=ampFile)

## Classify the droplets into 4 clusters.
aWell <- kmeansClassify(aWell, centres=4)

## Write the amplitudes to a CSV file with the old and new classifications.
exportTable(aWell,
  location=file.path(tmpOut, "With_Kmeans_B03_Amplitude.csv"))

## Write the amplitudes to a CSV file with the new classification only.
exportTable(aWell,
  location=file.path(tmpOut, "With_Kmeans_B03_Amplitude.csv"),
  cMethod="kmeans")

## Read all amplitude files in a directory to a ddpcrPlate object.
moreAmpsDir <- system.file("extdata", "more-amplitudes", package="twoddpcr")
krasPlate <- ddpcrPlate(wells=moreAmpsDir)

## Classify the droplets into 4 clusters.
krasPlate <- kmeansClassify(krasPlate, centres=4)

## Write the amplitudes to multiple files in a directory with the old and
## new classifications.
exportTable(krasPlate, location=file.path(tmpOut, "amplitudes-classified"))

## Write the amplitudes to multiple files with the new classification only
## and a custom prefix for the filenames.
exportTable(krasPlate, location=file.path(tmpOut, "amplitudes-classified"),
            cMethod="kmeans", prefix="Kmeans_Only_")

## Export to a zip file.
```

```
exportZip(krasPlate,
          location=file.path(tmpOut, "amplitudes-classified/all.zip"),
          cMethod="kmeans", prefix="Kmeans_Only_")
```

`extractPlateName` *Try to get plate name from a filename.*

Description

If the given filename is of the form "<PlateName>_<WellName>_Amplitude.csv", where <WellName> is of the form A01, B01, etc., then this function can extract the <PlateName> component. Otherwise, an empty string is returned.

Usage

```
extractPlateName(filename)
```

Arguments

`filename`	A character string corresponding to a filename with .csv extension.

Value

A character string corresponding to the plate name. This is "" if filename is not in a known format.

Author(s)

Anthony Chiu, <anthony.chiu@cruk.manchester.ac.uk>

Examples

```
## Get the plate name (recognised format).
extractPlateName(c("Sample_Plate_Name_G02_Amplitude.csv"))

## Get the plate name (unrecognised format).
extractPlateName(c("Sample_G02.csv"))
```

`extractWellNames` *Try to get well names from a vector of filenames.*

Description

If each of the given filenames are of the form "<PlateName>_<WellName>_Amplitude.csv", where <WellName> is of the form A01, B01, etc., then this function can extract the <WellName> component. Otherwise, the whole file name is assumed to be the well name.

Usage

```
extractWellNames(filenames)
```

Arguments

`filenames`	A character vector of filenames with .csv extension.

Value

A character vector of well names.

Author(s)

Anthony Chiu, <anthony.chiu@cruk.manchester.ac.uk>

Examples

```
## Get the well names (recognised format).
extractWellNames(c("Sample_Plate_Name_G02_Amplitude.csv",
                   "Sample_Plate_Name_H02_Amplitude.csv",
                   "Sample_Plate_Name_A03_Amplitude.csv",
                   "Sample_Plate_Name_B03_Amplitude.csv"))

## Get the well names (unrecognised format).
extractWellNames(c("Sample_G02.csv",
                   "Sample_H02.csv",
                   "Sample_A03.csv",
                   "Sample_B03.csv"))
```

facetPlot *Draw each of the individual wells in a ddPCR experiment.*

Description

Plot each of the wells in a ddpcrPlate object or a large data frame of droplets. By default, a density plot is returned for speed purposes.

Usage

```
facetPlot(
  droplets,
  ch1Label = "Ch1 Amplitude",
  ch2Label = "Ch2 Amplitude",
  cMethod = NULL,
  binwidth = 100,
  pointSize = 0.1,
  plotLimits = list(x = c(1000, 9000), y = c(3000, 13500)),
  showEmptyWells = FALSE
)

## S4 method for signature 'data.frame'
facetPlot(
  droplets,
  ch1Label = "Ch1 Amplitude",
  ch2Label = "Ch2 Amplitude",
  cMethod = NULL,
  binwidth = 100,
  pointSize = 0.1,
  plotLimits = list(x = c(1000, 9000), y = c(3000, 13500)),
  showEmptyWells = FALSE
)

## S4 method for signature 'ddpcrPlate'
facetPlot(
  droplets,
  ch1Label = "Ch1 Amplitude",
  ch2Label = "Ch2 Amplitude",
  cMethod = NULL,
  binwidth = 100,
  pointSize = 0.1,
  plotLimits = list(x = c(1000, 9000), y = c(3000, 13500)),
  showEmptyWells = FALSE
)
```

Arguments

droplets	A ddpcrPlate object or a data frame of droplet amplitudes with a "Well" column.
ch1Label	The label for the channel 1 target. Defaults to "Ch1 Amplitude".
ch2Label	The label for the channel 2 target. Defaults to "Ch2 Amplitude".
cMethod	This should be the name or column number of droplets corresponding to the classification to be plotted. This column should only have entries in "NN", "PN", "NP, "PP", "Rain" and "N/A". If "None", plots the droplets with all of them classified as N/A. If NULL, a density plot is plotted. Defaults to NULL.
binwidth	The width of each hexagonal bin in the density plot. Ignored if cMethod is not NULL (see pointSize instead). Defaults to 100.
pointSize	If cMethod is not NULL, this is the size to draw each droplet. Otherwise this parameter is ignored (see binwidth instead). Defaults to 0.1.
plotLimits	A list of 2-element vectors with names x and y. These are used to fix the x and y limits of the plot, which is especially useful for comparing plots. Defaults to list(x=c(1000,9000), y=c(3000, 13500)).
showEmptyWells	If TRUE, plots a facet_grid of all the wells in the plate, including the empty ones. If FALSE, plots a facet_wrap of only the loaded (nonempty) wells. Defaults to FALSE.

Value

A collection of plots as a ggplot object.

Author(s)

Anthony Chiu, <anthony.chiu@cruk.manchester.ac.uk>

References

The nice log-scaled palette was achieved using http://www.everydayanalytics.ca/2014/09/5-ways-to-do-2d-histograms-in-r.html

See Also

By default, each subplot uses the same plotting style as heatPlot.

Examples

```
## Plot a facet wrap of density plots of each well.
krasPlate <- ddpcrPlate(wells=KRASdata)
facetPlot(krasPlate)
```

`flatPlot` *Plot droplet amplitudes with all droplets classified as "N/A" (or a chosen class).*

Description

There are occasions where classification algorithms fail for various reasons (such as poor choice/number of centres in k-means clustering). In these cases, it may be helpful to the user if an app draws a 'flat' plot with just one colour rather than nothing at all.

If a `ddpcrWell` object is given as a parameter, plot the droplets in the well and colour them according to a given class.

If a `ddpcrPlate` object is given as a parameter, plot the droplets in all wells and colour them according to a given class.

Usage

```
flatPlot(
  droplets,
  ch1Label = "Ch1 Amplitude",
  ch2Label = "Ch2 Amplitude",
  classString = ddpcr$na,
  initialCentres = NULL,
  selectedCentre = NULL,
  plotLimits = list(x = c(1000, 9000), y = c(3000, 13500))
)

## S4 method for signature 'data.frame'
flatPlot(
  droplets,
  ch1Label = "Ch1 Amplitude",
  ch2Label = "Ch2 Amplitude",
  classString = ddpcr$na,
  initialCentres = NULL,
  selectedCentre = NULL,
  plotLimits = list(x = c(1000, 9000), y = c(3000, 13500))
)

## S4 method for signature 'ddpcrWell'
flatPlot(
  droplets,
  ch1Label = "Ch1 Amplitude",
  ch2Label = "Ch2 Amplitude",
  classString = ddpcr$na,
  initialCentres = NULL,
  selectedCentre = NULL,
  plotLimits = list(x = c(1000, 9000), y = c(3000, 13500))
```

```
)

## S4 method for signature 'ddpcrPlate'
flatPlot(
  droplets,
  ch1Label = "Ch1 Amplitude",
  ch2Label = "Ch2 Amplitude",
  classString = ddpcr$na,
  initialCentres = NULL,
  selectedCentre = NULL,
  plotLimits = list(x = c(1000, 9000), y = c(3000, 13500))
)
```

Arguments

droplets	A data frame of droplet amplitudes, or a ddpcrWell or ddpcrPlate object.
ch1Label	The label for the channel 1 target. Defaults to "Ch1 Amplitude".
ch2Label	The label for the channel 2 target. Defaults to "Ch2 Amplitude".
classString	The class that all droplets should be classified as. Defaults to the ddpcr$na ("N/A") character string.
initialCentres	A data frame of initial centres to plot (e.g. initial cluster centres used in the k-means). This is _not_ restricted to the class classString only. If NULL, nothing is plotted. Defaults to NULL.
selectedCentre	An initial centre to highlight. This should be either "NN", "PN", "NP" or "PP", but is _not_ restricted to the class 'classString' only. If NULL, nothing is highlighted. Defaults to NULL.
plotLimits	A list of 2-element vectors with names x and y. These are used to fix the x and y limits of the plot, which is especially useful for comparing plots. Defaults to list(x=c(1000, 9000), y=c(3000, 13500)).

Value

A ggplot object in just one colour corresponding to classString.

Author(s)

Anthony Chiu, <anthony.chiu@cruk.manchester.ac.uk>

Examples

```
## Plot the data frame with no classification.
aWell <- KRASdata[["E03"]]
flatPlot(aWell)

## Take a ddpcrWell object that is mostly classified as "NN" and colour it
## as such.
aWell <- KRASdata[["H04"]]
emptiedWell <- aWell[aWell$Cluster == 1, ]
```

```
emptiedWell <- ddpcrWell(well=emptiedWell)
flatPlot(emptiedWell, classString="NN")

## Plotting all of a ddpcrPlate object works the same way.
krasPlate <- ddpcrPlate(wells=KRASdata)
flatPlot(krasPlate)
```

`fullCopiesSummary` *Get all of the counts data in a summarised data frame.*

Description

Returns a data frame with all the copies information, plus any optional columns. This function is intended to be used as a final summary of the molecule counts.

Usage

```
fullCopiesSummary(df, extraCols = NULL)
```

Arguments

`df`	A data frame generated by `fullCountsSummary`.
`extraCols`	A vector of column names from `df` to include. If `NULL`, no extra columns are added. Defaults to `NULL`.

Value

A data frame with the basic figures, the mutant counts, the wild type counts, the summarised counts, and extraCols if specified. Prints an additional column for notes, indicating whether this run failed or if there were fewer than 8000 accepted droplets.

Author(s)

Anthony Chiu, <anthony.chiu@cruk.manchester.ac.uk>

Examples

```
df <- fullCountsSummary(KRAScountsQS)
fullCopiesSummary(df)
```

`fullCountsSummary` *Take a data frame and compute the abundance of molecules.*

Description

Returns a data frame with basic counts, the concentration of each kind of target molecule, the ratio ch1:ch2 molecules, and the fractional abundance of ch1 molecules in the overall count.

Usage

```
fullCountsSummary(
  df,
  ch1Label = "Mt",
  ch2Label = "Wt",
  rows = NULL,
  rowID = NULL,
  keepCols = NULL,
  keepColNames = NULL
)
```

Arguments

`df`	A data frame with droplet count columns in one of the following formats: • `PP`, `PN`, `NP`, `NN`; • `Ch1.Ch2.`, `Ch1.Ch2..1`, `Ch1.Ch2..2`, `Ch1.Ch2..3`; • `Ch1+Ch2+`, `Ch1+Ch2-`, `Ch1-Ch2+`, `Ch1-Ch2-`; or • `Ch1pCh2p`, `Ch1pCh2n`, `Ch1nCh2p`, `Ch1nCh2n`.
`ch1Label`	The prefix to use for the channel 1 target. Defaults to "Mt".
`ch2Label`	The prefix to use for the channel 2 target. Defaults to "Wt".
`rows`	A vector of rows (numbers or well names) to keep from the original data frame. If set to `NULL`, all wells will be used. Defaults to `NULL`.
`rowID`	If set, this field is used as the row names. If `NULL`, the existing row names from `df` are used. Defaults to `NULL`.
`keepCols`	A vector of columns to keep from `df`. If `NULL`, no extra columns are added. Defaults to `NULL`.
`keepColNames`	A vector of new column names for `keepCols`. If `NULL`, the column names from `keepCols` are reused. Defaults to `NULL`.

Value

A data frame with

- `rowID` as the row names (if given);
- the droplet counts per channel;

- the number of ch1 and ch2 positive and negative readings;
- the ch1 and ch2 concentration, copies per 20ul, and total copies per 20ul;
- the ratio of ch1 to ch2 molecules; and
- the fractional abundance of ch1 molecules in the overall molecule count (as a percentage).

Author(s)

Anthony Chiu, <anthony.chiu@cruk.manchester.ac.uk>

Examples

```
## Take a data frame with row names given by the well names. Get estimates
## for the numbers of molecules in each sample..
fullCountsSummary(KRAScounts)

## Keep only the row numbers 3, 6 and 9.
fullCountsSummary(KRAScounts, rows=c(3, 6, 9))

## Keep only the rows labelled "F03", "A04", "D04".
fullCountsSummary(KRAScounts, rows=c("F03", "A04", "D04"))

## Take a data frame with a 'Well' column and do the same as above.
fullCountsSummary(KRAScountsWellCol, rowID="Well")

## Keep the 'InputAmount' column.
fullCountsSummary(KRAScounts, keepCols=c("InputAmount"))

## Keep the 'InputAmount' column and rename it.
fullCountsSummary(KRAScounts, keepCols=c("InputAmount"),
                  keepColNames=c("NanogramsIn"))
```

`getCutOff` *Find the standard deviation of droplets (in a given class) multipied by a given constant.*

Description

For a specified class, take a data frame of droplet amplitudes and compute the standard deviation multiplied by a level of accuracy.

Usage

```
getCutOff(droplets, cl, level = 3, classCol = "class")
```

Arguments

droplets	A data frame of droplets with "Ch1.Amplitude" and "Ch2.Amplitude" columns, as well as a class column (see classCol).
cl	The class to focus on. Typically one of "NN", "PN", "NP" and "PP".
level	A constant by which we will multiply the standard deviation. Defaults to 5.
classCol	The column (name or number) from 'droplets' representing the class.

Value

A list with named elements 'ch1' and 'ch2', each giving the error bound for the corresponding channel. If the number of droplets is either 0 or 1, return list(ch1=0, ch2=0) to avoid any errors. Otherwise, return level * (sd of the droplets in each channel).

Author(s)

Anthony Chiu, <anthony.chiu@cruk.manchester.ac.uk>

---

ggplot.well *ggplot methods for the* ddpcrWell *and* ddpcrPlate *classes.*

Description

These functions work in the same way as the original ggplot method, but handles the coercion of the object into a data frame.

ggplot.well is a ggplot method for the ddpcrWell class.

ggplot.multiwell is a ggplot method for the ddpcrPlate class.

Usage

```
ggplot.well(
  data,
  mapping = aes_string(x = "Ch2.Amplitude", y = "Ch1.Amplitude", colour = cMethod),
  cMethod = NULL,
  ...,
  environment = parent.frame()
)

## S4 method for signature 'ddpcrWell'
ggplot.well(
  data,
  mapping = aes_string(x = "Ch2.Amplitude", y = "Ch1.Amplitude", colour = cMethod),
  cMethod = "None",
  ...,
  environment = parent.frame()
)
```

```
ggplot.plate(
  data,
  mapping = aes_string(x = "Ch2.Amplitude", y = "Ch1.Amplitude", colour = class),
  cMethod = "None",
  ...,
  environment = parent.frame()
)

## S4 method for signature 'ddpcrPlate'
ggplot.plate(
  data,
 mapping = aes_string(x = "Ch2.Amplitude", y = "Ch1.Amplitude", colour = cMethod),
  cMethod = "None",
  ...,
  environment = parent.frame()
)
```

Arguments

data	A ddpcrWell or ddpcrPlate object.
mapping	A list of aesthetic mappings to use for the plot. Defaults to ggplot2::aes_string(x="Ch2.Amplitude", y="Ch1.Amplitude",colour=cMethod), where cMethod is taken from the parameter of the same name.
cMethod	The name or column number of the classification to use. This is renamed internally to "class" for use with mapping. Defaults to "None".
...	Other arguments passed onto ggplot.
environment	Where to look if a mapping variable is not defined. Defaults to parent.frame(), i.e. the environment in which ggplot.well() or ggplot.multiwell() is called.

Value

A ggplot object using the slots in the given object.

Author(s)

Anthony Chiu, <anthony.chiu@cruk.manchester.ac.uk>

See Also

dropletPlot builds upon these ggplot methods to plot droplet amplitude plots with a colour-blind friendly palette.

The original ggplot method in the ggplot2 package is used internally.

Examples

```
## Plot the droplets in one well.
library(ggplot2)
```

```
aWell <- ddpcrWell(KRASdata[["E03"]])
ggplot.well(aWell, cMethod="Cluster") + geom_point()

## Plot the droplets in all of the wells in a single plot.
krasPlate <- ddpcrPlate(KRASdata)
ggplot.plate(krasPlate, cMethod="Cluster") + geom_point()
```

`gridClassify` *Use a 'grid' to create training data for classification algorithms.*

Description

Classify droplets as "NN", "NP", "PN" or "PP". The classification is based on upper bounds for negative readings and lower bounds for positive readings; see the details and parameters for more detail. If required (see the `trainingData` parameter), droplets that are not classified will be given the label "N/A".

Usage

```
gridClassify(
  droplets,
  ch1NNThreshold = 6500,
  ch2NNThreshold = 1900,
  ch1NPThreshold = 6500,
  ch2NPThreshold = 5000,
  ch1PNThreshold = 10000,
  ch2PNThreshold = 2900,
  ch1PPThreshold = 7500,
  ch2PPThreshold = 5000,
  ...
)

## S4 method for signature 'data.frame'
gridClassify(
  droplets,
  ch1NNThreshold = 6500,
  ch2NNThreshold = 1900,
  ch1NPThreshold = 6500,
  ch2NPThreshold = 5000,
  ch1PNThreshold = 10000,
  ch2PNThreshold = 2900,
  ch1PPThreshold = 7500,
  ch2PPThreshold = 5000,
  trainingData = TRUE,
  fullTable = TRUE,
  naLabel = ddpcr$rain
```

```
)

## S4 method for signature 'ddpcrWell'
gridClassify(
  droplets,
  ch1NNThreshold = 6500,
  ch2NNThreshold = 1900,
  ch1NPThreshold = 6500,
  ch2NPThreshold = 5000,
  ch1PNThreshold = 10000,
  ch2PNThreshold = 2900,
  ch1PPThreshold = 7500,
  ch2PPThreshold = 5000,
  classMethodLabel = "grid",
  naLabel = ddpcr$rain
)

## S4 method for signature 'ddpcrPlate'
gridClassify(
  droplets,
  ch1NNThreshold = 6500,
  ch2NNThreshold = 1900,
  ch1NPThreshold = 6500,
  ch2NPThreshold = 5000,
  ch1PNThreshold = 10000,
  ch2PNThreshold = 2900,
  ch1PPThreshold = 7500,
  ch2PPThreshold = 5000,
  classMethodLabel = "grid",
  naLabel = ddpcr$rain
)
```

Arguments

`droplets`	A `ddpcrWell` object or a data frame of droplet amplitudes with columns `Ch1.Amplitude` and `Ch2.Amplitude`.
`ch1NNThreshold`	The channel 1 upper bound for the NN class. Defaults to 6500.
`ch2NNThreshold`	The channel 2 upper bound for the NN class. Defaults to 1900.
`ch1NPThreshold`	The channel 1 upper bound for the NP class. Defaults to 6500.
`ch2NPThreshold`	The channel 2 lower bound for the NP class. Defaults to 5000.
`ch1PNThreshold`	The channel 1 lower bound for the PN class. Defaults to 10000.
`ch2PNThreshold`	The channel 2 upper bound for the PN class. Defaults to 2900.
`ch1PPThreshold`	The channel 1 lower bound for the PP class. Defaults to 7500.
`ch2PPThreshold`	The channel 2 lower bound for the PP class. Defaults to 5000.
`...`	Other options depending on the type of `droplets`.

`trainingData` Whether to use the output as training data. If `TRUE`, returns the _full table_ with the "N/A" entries removed; if `FALSE`, the "N/A" entries are retained. Taken to be `FALSE` if `fullTable` is set to `FALSE`. Defaults to `TRUE`. Ignored if `droplets` is not a data frame.

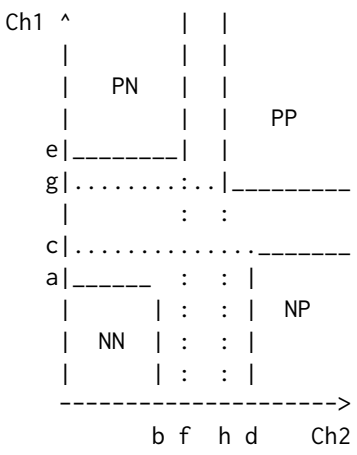
`fullTable` Whether to return a data frame including amplitude figures. If `TRUE`, a data frame with columns `Ch1.Amplitude`, `Ch1.Amplitude` and `class` is returned. If `FALSE`, a factor with levels in `ddpcr$classesRain` is returned, where each entry corresponds to each row in `droplets` (and `trainingData` is automatically set to `FALSE`). Defaults to `TRUE`. Ignored if `droplets` is not a data frame.

`naLabel` The label to use for unclassified droplets. Should be either ddpcr$na ("N/A") or ddpcr$rain ("Rain"). Defaults to ddpcr$rain.

`classMethodLabel`
A name (as a character string) of the classification method. Defaults to "grid".

Details

The `threshold` parameters correspond to those in the following diagram:



Specifically:

a: `ch1NNThreshold`,

b: `ch2NNThreshold`,

c: `ch1PNThreshold`,

d: `ch2PNThreshold`,

e: `ch1NPThreshold`,

f: `ch2NPThreshold`,

g: `ch1PPThreshold`,

h: `ch2PPThreshold`.

Value

If `droplets` is a data frame, return a data frame or factor (depending on the `trainingData` and `fullTable` parameters) with a classification for droplets in the chosen regions.

If `droplets` is a `ddpcrWell` object, return a `ddpcrWell` object with the appropriate classification.

If `droplets` is a `ddpcrPlate` object, return a `ddpcrPlate` object with the appropriate classification.

Author(s)

Anthony Chiu, <anthony.chiu@cruk.manchester.ac.uk>

See Also

`thresholdClassify` is a special case of this function.

`removeDropletClasses` retrieves a data frame with the "N/A" (and "Rain") droplets removed. This can used for transforming a grid-like classification into usable training data.

Examples

```
## Use a grid to set training data for a data frame.
sgCl <- gridClassify(KRASdata[["E03"]],
                     ch1NNThreshold=5700, ch2NNThreshold=1700,
                     ch1NPThreshold=5400, ch2NPThreshold=5700,
                     ch1PNThreshold=9700, ch2PNThreshold=2050,
                     ch1PPThreshold=7200, ch2PPThreshold=4800)
str(sgCl)

## For data frame only, we can set the trainingData flag to FALSE so that
## the unclassified droplets are retained but labelled as "N/A"
sgCl <- gridClassify(KRASdata[["E03"]],
                     ch1NNThreshold=5700, ch2NNThreshold=1700,
                     ch1NPThreshold=5400, ch2NPThreshold=5700,
                     ch1PNThreshold=9700, ch2PNThreshold=2050,
                     ch1PPThreshold=7200, ch2PPThreshold=4800,
                     trainingData=FALSE)
dropletPlot(sgCl, cMethod="class")

## The same works for ddpcrWell objects.
aWell <- ddpcrWell(well=KRASdata[["E03"]])
aWell <- gridClassify(aWell,
                      ch1NNThreshold=5700, ch2NNThreshold=1700,
                      ch1NPThreshold=5400, ch2NPThreshold=5700,
                      ch1PNThreshold=9700, ch2PNThreshold=2050,
                      ch1PPThreshold=7200, ch2PPThreshold=4800)
str(aWell)

## ddpcrPlate objects work in exactly the same way.
krasPlate <- ddpcrPlate(wells=KRASdata)
krasPlate <- gridClassify(krasPlate)
lapply(plateClassification(krasPlate, withAmplitudes=TRUE), head, n=1)
```

```
## The default classification method (column name) is 'gridClassify',
## which may be a bit long. It can be changed.
krasPlate <- gridClassify(krasPlate, classMethodLabel="training")
lapply(plateClassification(krasPlate, withAmplitudes=TRUE), head, n=1)
```

`heatPlot` *Draw a heat plot of the droplets.*

Description

Using alpha transparency only, it is generally difficult to see where droplets are truly distributed and concentrated. A heat (density) plot gives a better illustration of this.

Usage

```
heatPlot(
  droplets,
  ch1Label = "Ch1 Amplitude",
  ch2Label = "Ch2 Amplitude",
  binwidth = 100,
  plotLimits = list(x = c(1000, 9000), y = c(3000, 13500))
)

heatPlot(
  droplets,
  ch1Label = "Ch1 Amplitude",
  ch2Label = "Ch2 Amplitude",
  binwidth = 100,
  plotLimits = list(x = c(1000, 9000), y = c(3000, 13500))
)

## S4 method for signature 'data.frame'
heatPlot(
  droplets,
  ch1Label = "Ch1 Amplitude",
  ch2Label = "Ch2 Amplitude",
  binwidth = 100,
  plotLimits = list(x = c(1000, 9000), y = c(3000, 13500))
)

## S4 method for signature 'ddpcrWell'
heatPlot(
  droplets,
  ch1Label = "Ch1 Amplitude",
  ch2Label = "Ch2 Amplitude",
  binwidth = 100,
```

```
  plotLimits = list(x = c(1000, 9000), y = c(3000, 13500))
)

## S4 method for signature 'ddpcrPlate'
heatPlot(
  droplets,
  ch1Label = "Ch1 Amplitude",
  ch2Label = "Ch2 Amplitude",
  binwidth = 100,
  plotLimits = list(x = c(1000, 9000), y = c(3000, 13500))
)
```

Arguments

droplets	A data frame of droplet amplitudes, a ggplot, ddpcrWell or ddpcrPlate object.
ch1Label	The label for the channel 1 target. Defaults to "Ch1 Amplitude".
ch2Label	The label for the channel 2 target. Defaults to "Ch2 Amplitude".
binwidth	The width of each hexagonal bin in the 2d heat (density) plot. Defaults to 100.
plotLimits	A list of 2-element vectors with names x and y. These are used to fix the x and y limits of the plot, which is especially useful for comparing plots. Defaults to list(x=c(1000,9000), y=c(3000, 13500)).

Value

A heat plot as a ggplot object.

Author(s)

Anthony Chiu, <anthony.chiu@cruk.manchester.ac.uk>

References

The nice log-scaled palette was achieved using http://www.everydayanalytics.ca/2014/09/5-ways-to-do-2d-histograms-in-r.html

Examples

```
## Density plot of a data frame.
heatPlot(KRASdata[["E03"]])

## Density plot of a ddpcrWell object.
aWell <- ddpcrWell(well=KRASdata[["E03"]])
heatPlot(aWell)

## Density plot of a ddpcrPlate object with an adjusted bin size.
krasPlate <- ddpcrPlate(wells=KRASdata)
heatPlot(krasPlate, binwidth=50)
```

isEmpty,ddpcrWell-method *Is a* ddpcrWell *object empty?*

Description

Returns a logical value as to whether the given object has no droplets/wells.

Usage

```
## S4 method for signature 'ddpcrWell'
isEmpty(x)

## S4 method for signature 'ddpcrPlate'
isEmpty(x)
```

Arguments

x	An object to test for emptiness.

Value

A logical value.

Author(s)

Anthony Chiu, <anthony.chiu@cruk.manchester.ac.uk>

Examples

```
## Check that an empty ddpcrPlate object is in fact empty.
krasPlate <- ddpcrPlate(wells=list())
isEmpty(krasPlate)

## Now add some amplitude data and check that is not empty.
krasPlate <- ddpcrPlate(wells=KRASdata)
isEmpty(krasPlate)
```

kmeansClassify *K-means classify the wells in a* ddpcrWell *or* ddpcrPlate *object, or in a data frame.*

Description

If droplets is a data frame, the droplets are classified using the k-means clustering algorithm.

For ddpcrWell, the droplets are classified by using the k-means clustering algorithm.

For ddpcrPlate, all of the wells are combined and classified, with this new classification assigned to the ddpcrPlate object.

Usage

```
kmeansClassify(
  droplets,
  centres = matrix(c(0, 0, 10000, 0, 0, 7000, 10000, 7000), ncol = 2, byrow = TRUE),
  ...
)

## S4 method for signature 'data.frame'
kmeansClassify(
  droplets,
  centres = matrix(c(0, 0, 10000, 0, 0, 7000, 10000, 7000), ncol = 2, byrow = TRUE),
  fullTable = TRUE
)

## S4 method for signature 'ddpcrWell'
kmeansClassify(
  droplets,
  centres = matrix(c(0, 0, 10000, 0, 0, 7000, 10000, 7000), ncol = 2, byrow = TRUE)
)

## S4 method for signature 'ddpcrPlate'
kmeansClassify(
  droplets,
  centres = matrix(c(0, 0, 10000, 0, 0, 7000, 10000, 7000), ncol = 2, byrow = TRUE)
)
```

Arguments

droplets	A ddpcrWell or ddpcrPlate object, or a data frame with columns Ch1.Amplitude and Ch2.Amplitude.
centres	Either: • A matrix corresponding to the initial centres to use for the k-means algorithm; or

- An integer corresponding to the number of clusters. If this is set, the initial centres are randomly set.

Defaults to `matrix(c(0, 0, 10000, 0, 0, 7000, 10000, 7000), ncol=2,byrow=TRUE)`

`...` Other options depending on the type of `droplets`.

`fullTable` If `TRUE`, returns a full data frame of droplets and their classification; if `FALSE`, simply returns a factor corresponding to this classification. Defaults to `TRUE`.

Value

An object with the new classification.

If `droplets` is a data frame, a list is returned with the following components:

`data` A data frame or vector corresponding to the classification.

`centres` A data frame listing the final centre points from the k-means algorithm with the corresponding cluster labels.

Author(s)

Anthony Chiu, <anthony.chiu@cruk.manchester.ac.uk>

See Also

This method uses the `kmeans` function.

To manually set and retrieve classifications, use the `wellClassification`, `plateClassification` and `plateClassificationMethod` methods.

For a supervised classification approach, one may want to consider `knnClassify`.

Examples

```
### Use the KRASdata dataset for all of these examples.

## Use K-means clustering to classify droplets into four (the default
## number) classes.
aWell <- kmeansClassify(KRASdata[["E03"]])

## We can look the the classification or the centres.
head(aWell$data)
aWell$centres

## Specify 3 centres for a different sample in KRASdata.
aWell <- kmeansClassify(KRASdata[["H04"]], centres=3)
head(aWell$data)

## We can be more specific with the choice of centres.
aWell <- kmeansClassify(KRASdata[["H04"]],
                        centres=matrix(c(5000, 1500, 5500, 7000, 10000,
                                         2000), ncol=2, byrow=TRUE))

## We can use \code{ddpcrWell} objects directly as a parameter.
```

```
aWell <- ddpcrWell(well=KRASdata[["E03"]])
kmeansClassify(aWell)

## We can take multiple samples in a \code{ddpcrPlate} object and
## classify everything together.
krasPlate <- ddpcrPlate(wells=KRASdata)
kmeansClassify(krasPlate)
```

`knnClassify` *Use the k-nearest neighbour algorithm to classify the wells in a* `ddpcrWell` *or* `ddpcrPlate` *object, or in a data frame.*

Description

If `droplets` is a data frame, the droplets are classified using the k-nearest neighbour algorithm against a training data set.

If `droplets` is a `ddpcrWell` object, the droplets in the well are classified and returned in another `ddpcrWell` object.

If `droplets` is a `ddpcrPlate` object, the wells are combined and classified together, with the resulting classification assigned to the `ddpcrPlate` object.

Usage

```
knnClassify(droplets, trainData, cl, k, prob = 0, ...)

## S4 method for signature 'data.frame'
knnClassify(droplets, trainData, cl, k, prob = 0, fullTable = TRUE)

## S4 method for signature 'ddpcrWell'
knnClassify(droplets, trainData, cl, k, prob = 0)

## S4 method for signature 'ddpcrPlate'
knnClassify(droplets, trainData, cl, k, prob = 0)
```

Arguments

`droplets`	A `ddpcrWell` or `ddpcrPlate` object, or a data frame with columns `Ch1.Amplitude` and `Ch2.Amplitude`.
`trainData`	A data frame of training data with columns `Ch1.Amplitude` and `Ch2.Amplitude`.
`cl`	A vector of classes corresponding to `trainData`.
`k`	The number of nearest neighbours to use in the algorithm.
`prob`	The minimal proportion of votes for the winning class needed to assert that a droplet belongs to the class. This figure should be a float between 0 and 1. For example, if 0.6 then at least 60 k-nearest neighbours need to be of one class, otherwise it is classified as "Rain". Defaults to 0, i.e. we do not use "Rain".

...	Other options depending on the type of droplets.
fullTable	If TRUE, returns a full data frame of droplets and their classification; if FALSE, simply returns the classified vector. Defaults to TRUE.

Value

An object with the new classification.

If droplets is a data frame, return data frame or factor (depending on the value of fullTable) of the droplet classification under the k-NN algorithm.

Author(s)

Anthony Chiu, <anthony.chiu@cruk.manchester.ac.uk>

See Also

This method uses the knn function.

To manually set and retrieve classifications, use the wellClassification, plateClassification and plateClassificationMethod methods.

kmeansClassify provides a wrapper for the k-means clustering algorithm.

Examples

```
### Use the KRASdata dataset for all of these examples.

## Use k-means clustering to classify one sample. Use this as training
## data for the K-Nearest Neighbour algorithm.
trainingData <- KRASdata[["E03"]]
trainingData <- kmeansClassify(trainingData)$data

## Classify a dataframe using k-NN with k = 1 and the above training data.
aWell <- knnClassify(
           KRASdata[["F03"]],
           trainData=trainingData[, c("Ch1.Amplitude", "Ch2.Amplitude")],
           cl=trainingData$class,
           k=1)
dropletPlot(aWell, cMethod="class")  # visualising the classification

## We can change k to a larger number, here with a ddpcrWell object.
aWell <- ddpcrWell(well=KRASdata[["E03"]])
aWell <- knnClassify(
           aWell,
           trainData=trainingData[, c("Ch1.Amplitude", "Ch2.Amplitude")],
           cl=trainingData$class,
           k=3)
dropletPlot(aWell, cMethod="knn")  # visualising the classification

## Changing the 'prob' parameter means that droplets with less than 'prob'
## of the votes will not be classified. We do this for a ddpcrPlate
## object.
```

```
krasPlate <- ddpcrPlate(wells=KRASdata[c("E03", "H03", "C04", "F04")])
krasPlate <- knnClassify(
               krasPlate,
               trainData=trainingData[, c("Ch1.Amplitude", "Ch2.Amplitude")],
               cl=trainingData$class,
               k=3,
           prob=0.6)
dropletPlot(krasPlate, cMethod="knn")  # visualising the classification
```

KRAScounts *KRAS mutant and wild type droplet counts and Poisson estimates.*

Description

A data frame of droplet counts using the standard `Cluster` classification from `KRASdata`. Each row corresponds to a well/sample and the columns `PP`, `PN`, `NP` and `NN` show how many droplets were in each cluster. The remaining rows were calculated from the counts figures except for the `InputAmount` column.

`KRAScountsWellCol` is the same data frame but has a `Well` column instead of named rows.

`KRAScountsQS` is a data frame imported from a CSV created by Bio-Rad's QuantaSoft.

Usage

```
data(KRAScounts)

data(KRAScountsWellCol)

data(KRAScountsQS)
```

Format

A data frame where each row corresponds to a well/sample.

An object of class `data.frame` with 12 rows and 18 columns.

An object of class `data.frame` with 24 rows and 63 columns.

Details

The data frame was created by:

```
krasPlate <- ddpcrPlate(wells=KRASdata)
KRAScounts <- plateSummary(krasPlate, cMethod="Cluster")
onesVector <- c(1, 1, 1)
runAmount <- c(64 * onesVector, 16 * onesVector, 4 * onesVector, onesVector)
KRAScounts$InputAmount <- runAmount

KRAScountsWellCol <- KRAScounts
```

```
KRAScountsWellCol$Well <- rownames(KRAScounts)
KRAScountsWellCol <- KRAScountsWellCol[, c(18,1:17)]
rownames(KRAScountsWellCol) <- NULL
```

Value

A data frame.

Author(s)

From the KRASdata dataset created by Mahmood Ayub, <mahmood.ayub@cruk.manchester.ac.uk>

KRASdata *Droplet amplitude data for KRAS mutant and wild type molecules.*

Description

A data frame of ddPCR droplet amplitudes using KRAS cell lines and pre-defined mutant/wild type ratios. The existing classification under the column name `Cluster` was obtained from Bio-Rad's QuantaSoft by setting the `Ch1.Amplitude` threshold to `6789` and the `Ch2.Amplitude` threshold to `3000`.

Usage

```
data(KRASdata)
```

Format

A list of data frames, each of which has columns `Ch1.Amplitude`, `Ch2.Amplitude` and `Cluster`.

Value

A data frame.

Author(s)

From ddPCR experiments run by Mahmood Ayub, <mahmood.ayub@cruk.manchester.ac.uk>

mahalanobisRain *Define 'rain' (unclassified) droplets by fitting the clusters to bivariate normal distributions.*

Description

Assume that each of the classified clusters are bivariate normally distributed. We add fuzziness to the classifications by assigning droplets far away from the centres as "Rain". We use the Mahalanobis distance for each cluster to determine whether a droplet is 'too far away'.

Usage

```
mahalanobisRain(droplets, cMethod, maxDistances = 30, ...)

## S4 method for signature 'data.frame'
mahalanobisRain(droplets, cMethod, maxDistances = 30, fullTable = TRUE)

## S4 method for signature 'ddpcrWell'
mahalanobisRain(droplets, cMethod, maxDistances = 30)

## S4 method for signature 'ddpcrPlate'
mahalanobisRain(droplets, cMethod, maxDistances = 30)
```

Arguments

droplets	A ddpcrWell or ddpcrPlate object, or a data frame of droplets with "Ch1.Amplitude" and "Ch2.Amplitude" columns, as well as a class column.
cMethod	The name or column number of the classification for which we want to add rain to.
maxDistances	A list of (levels) with keys in c("NN", "PN","NP", "PP"). If the list is empty or set as NULL, no rain is added. If set as a single integer n, this value is taken for all classes, i.e. list("NN"=n, "PN"=n, "NP"=n, "PP"=n).
...	Other options depending on the type of droplets.
fullTable	If TRUE, returns a full data frame of droplets and their classification; if FALSE, simply returns a factor corresponding to this classification. Defaults to TRUE.

Value

An object where the specified class has "Rain" entries added.

Author(s)

Anthony Chiu, <anthony.chiu@cruk.manchester.ac.uk>

Examples

```
## Take a data frame of droplets of transform it into the rigth format.
droplets <- KRASdata[["E03"]]
droplets$Cluster <- relabelClasses(droplets, classCol="Cluster")

## Add rain as a new column.
droplets$ClusterMahRain <-
    mahalanobisRain(droplets, cMethod="Cluster", fullTable=FALSE)
table(droplets$ClusterMahRain)

## The maximum distance around each mean can be changed uniformly.
droplets$ClusterMahRain <-
    mahalanobisRain(droplets, cMethod="Cluster", maxDistances=35,
                    fullTable=FALSE)
table(droplets$ClusterMahRain)

## Or we can change the maximum distances for each individual cluster.
droplets$ClusterMahRain <-
    mahalanobisRain(droplets, cMethod="Cluster",
                    maxDistances=list(NN=35, NP=30, PN=30, PP=30),
                    fullTable=FALSE)
table(droplets$ClusterMahRain)

# This method works the same for ddpcrWell objects.
aWell <- ddpcrWell(well=KRASdata[["E03"]])
aWell <- mahalanobisRain(aWell, cMethod="Cluster")
table(wellClassification(aWell, cMethod="ClusterMahRain"))

# Likewise for ddpcrPlate objects.
krasPlate <- ddpcrPlate(wells=KRASdata[c("E03", "H03", "C04", "F04")])
krasPlate <- mahalanobisRain(krasPlate, cMethod="Cluster")
lapply(plateClassification(krasPlate, cMethod="ClusterMahRain"), table)
```

`mutantCopiesSummary` *Get the mutant copies per 20ul of a data frame in the context of the basic counts.*

Description

Returns a data frame with the basic figures and the number of mutant copies per 20ul.

Usage

```
mutantCopiesSummary(df)
```

Arguments

`df`	A data frame generated by `fullCountsSummary`.

Value

A data frame with the basic columns and mutant copies per 20ul.

Author(s)

Anthony Chiu, <anthony.chiu@cruk.manchester.ac.uk>

Examples

```
df <- fullCountsSummary(KRAScountsQS)
mutantCopiesSummary(df)
```

---

numDroplets *Retrieve the number of droplets.*

Description

Retrieves the number of droplets in a ddpcrWell or ddpcrPlate object.

Usage

```
numDroplets(theObject, ...)

## S4 method for signature 'ddpcrWell'
numDroplets(theObject)

## S4 method for signature 'ddpcrPlate'
numDroplets(theObject)
```

Arguments

theObject	A ddpcrPlate object.
...	Other parameters depending on the type of theObject.

Value

For ddpcrWell objects, return the number of droplets as an integer.

For ddpcrPlate objects, return a named vector. The names correspond to a well name and each item corresponding to the number of droplets in that well.

Author(s)

Anthony Chiu, <anthony.chiu@cruk.manchester.ac.uk>

Examples

```
## Count the number of droplets in a well.
aWell <- ddpcrWell(well=KRASdata[["E03"]])
numDroplets(aWell)

## Get all of the wells in a named vector.
krasPlate <- ddpcrPlate(wells=KRASdata)
(numberDroplets <- numDroplets(krasPlate))
sum(numberDroplets)

## We can choose to get a subset of the wells.
(numberDroplets <- numDroplets(krasPlate[c("H03", "A04")]))
sum(numberDroplets)
```

`numericInputRow` *Inline* `numericInputs`.

Description

By default, `numericInput` objects are added to the page below each other. This function allows for inline `numericInput` fields.

Usage

```
numericInputRow(inputId, label, value = "", size = NULL)
```

Arguments

`inputId`	The identifier for the numericInput object.
`label`	The text label to display alongside.
`value`	The default value of the numericInput.
`size`	The size (width) and maxlength.

Value

A `div` with the appropriate label and input field.

References

Adapted from: http://stackoverflow.com/a/21132918/1262569

`parseClusterCounts` *Retain cluster counts and user-specified columns in data frames.*

Description

Take a data frame of droplet counts and returns only the raw "PP", "PN", "NP" and "NN" counts, plus any additional columns specified.

Usage

```
parseClusterCounts(
  df,
  rows = NULL,
  rowID = NULL,
  keepCols = NULL,
  keepColNames = NULL
)
```

Arguments

`df`	A data frame with droplet count columns in one of the following formats: • `PP`, `PN`, `NP`, `NN`; • `Ch1.Ch2.`, `Ch1.Ch2..1`, `Ch1.Ch2..2`, `Ch1.Ch2..3`; • `Ch1+Ch2+`, `Ch1+Ch2-`, `Ch1-Ch2+`, `Ch1-Ch2-`; or • `Ch1pCh2p`, `Ch1pCh2n`, `Ch1nCh2p`, `Ch1nCh2n`.
`rows`	A vector of rows (numbers or well names) to keep from the original data frame. If set to `NULL`, all wells will be used. Defaults to `NULL`.
`rowID`	If set, this field is used as the row names. If `NULL`, the existing row names from `df` are used. Defaults to `NULL`.
`keepCols`	A vector of columns to keep from `df`. If `NULL`, no extra columns are added. Defaults to `NULL`.
`keepColNames`	A vector of new column names for `keepCols`. If `NULL`, the column names from `keepCols` are reused. Defaults to `NULL`.

Value

A data frame with the counts in the `PP`, `PN`, `PN` and `PN` convention.

Author(s)

Anthony Chiu, <anthony.chiu@cruk.manchester.ac.uk>

Examples

```
## Take a data frame with row names given by the well names. Get a simple
## count of droplets in each cluster.
parseClusterCounts(KRAScounts)

## Keep only the row numbers 3, 6 and 9.
parseClusterCounts(KRAScounts, rows=c(3, 6, 9))

## Keep only the rows labelled "F03", "A04", "D04".
parseClusterCounts(KRAScounts, rows=c("F03", "A04", "D04"))

## Take a data frame with a 'Well' column and do the same as above.
parseClusterCounts(KRAScountsWellCol, rowID="Well")

## Keep the 'InputAmount' column.
parseClusterCounts(KRAScounts, keepCols=c("InputAmount"))

## Keep the 'InputAmount' column and rename it.
parseClusterCounts(KRAScounts, keepCols=c("InputAmount"),
                   keepColNames=c("NanogramsIn"))
```

---

plateClassification *Set and retrieve classifications for multiple wells.*

Description

Retrieve multiple classification factors that have been assigned to a ddpcrPlate object.

Usage

```
plateClassification(
  theObject,
  cMethod = NULL,
  withAmplitudes = FALSE,
  wellCol = FALSE
)

## S4 method for signature 'ddpcrPlate'
plateClassification(
  theObject,
  cMethod = NULL,
  withAmplitudes = FALSE,
  wellCol = FALSE
)

plateClassification(theObject, cMethod) <- value
```

```
## S4 replacement method for signature 'ddpcrPlate,character,list'
plateClassification(theObject, cMethod) <- value

## S4 replacement method for signature 'ddpcrPlate,character,factor'
plateClassification(theObject, cMethod) <- value
```

Arguments

theObject	A ddpcrPlate object.
cMethod	This is the name of the classification to retrieve and should be a character vector. If NULL, then all of the classifications are obtained. Defaults to NULL.
withAmplitudes	If TRUE, the droplet amplitudes are included. Defaults to FALSE.
wellCol	If TRUE, an additional column is included in the output, where each entry is the name of the well from which the droplet originated. In this case, this setting forces the withAmplitudes parameter to TRUE. Defaults to FALSE.
value	Either: • A list of factors, where each item of the list corresponds to a well; • A single factor corresponding to all of the wells combined. This should be ordered by the order of the output of the amplitudes function when the rows of the data frames have been bound together, i.e. with do.call(rbind, amplitudes(theObject)).

Value

If requesting one classification without the amplitudes, a list of factors corresponding to the classifications is returned. Otherwise, a list of data frames is returned where each row corresponds to a droplet in the corresponding well.

Author(s)

Anthony Chiu, <anthony.chiu@cruk.manchester.ac.uk>

Examples

```
### The examples here show how this method works by setting classifications
### using data frames. To do this, we use the
### \code{\link{thresholdClassify}} method on _data frames_. Note that
### \code{thresholdClassify} also works directly on \code{ddpcrWell} and
### \code{ddpcrPlate} objects; this is simply an illustration of
### how to use the \code{plateClassification} method directly. In general,
### it is recommended to use \code{thresholdClassify} directly on
### \code{ddpcrPlate} objects.

## Create a ddpcrPlate object.
krasPlate <- ddpcrPlate(wells=KRASdata)

## Classify a data frame of droplets and keep it in a _single_ data frame.
## Set the new classification from this.
droplets <- do.call(rbind, amplitudes(krasPlate))
```

```
clSingle <- thresholdClassify(droplets,
                              ch1Threshold=7000, ch2Threshold=3500,
                              fullTable=FALSE)
plateClassification(krasPlate, "thresholdSing") <- clSingle

## We can also set the new classification from a list of factors.
clList <- lapply(KRASdata, thresholdClassify, ch1Threshold=7000,
                 ch2Threshold=3500, fullTable=FALSE)
plateClassification(krasPlate, "thresholdList") <- clList

## We can get all of the classifications as a list of data frames.
plate <- plateClassification(krasPlate)
lapply(plate, head, n=1)

## We can include the droplet amplitudes columns.
plate <- plateClassification(krasPlate, withAmplitudes=TRUE)
lapply(plate, head, n=1)

## We can focus on specific classifications.
plate <- plateClassification(krasPlate, cMethod=c("thresholdSing",
                                                  "thresholdList"))
lapply(plate, head, n=1)

## The wellCol option adds an extra column showing which well the droplet
## came from.
plate <- plateClassification(krasPlate, withAmplitudes=TRUE, wellCol=TRUE)
lapply(plate, head, n=1)
```

`plateClassificationMethod` *Set or retrieve the classification method strings for multiple wells.*

Description

`plateClassificationMethod` retrieves multiple classification methods that have been assigned to a `ddpcrPlate` object.

`commonClassificationMethod` retrieves the classification methods common to all the wells in the given `ddpcrPlate` object.

Usage

```
plateClassificationMethod(theObject)

## S4 method for signature 'ddpcrPlate'
plateClassificationMethod(theObject)

plateClassificationMethod(theObject, cMethod) <- value
```

```
## S4 replacement method for signature 'ddpcrPlate'
plateClassificationMethod(theObject, cMethod) <- value

commonClassificationMethod(theObject)

## S4 method for signature 'ddpcrPlate'
commonClassificationMethod(theObject)
```

Arguments

theObject	A ddpcrPlate object.
cMethod	This should represent existing classification method(s) for all wells in theObject. It can be given in the form of a: • Character vector. If this vector is shorter than the length of wells, this vector's elements will be repeated. • List of character vectors.
value	New classification method(s) in the same form as cMethod.

Value

plateClassificationMethod returns a list of character strings corresponding to the classification methods.

commonClassificationMethod returns a vector of character strings indicating which classification methods appear in all wells.

Author(s)

Anthony Chiu, <anthony.chiu@cruk.manchester.ac.uk>

See Also

plateClassification for the classifications.

Examples

```
## Get the classification methods for the KRASdata dataset.
krasPlate <- ddpcrPlate(wells=KRASdata)
plateClassificationMethod(krasPlate)

## Change the "Cluster" column to "QS".
plateClassificationMethod(krasPlate, "Cluster") <- "QS"
plateClassificationMethod(krasPlate)

## Usually, all of the classification names are the same for all wells. We
## use the \code{commonClassificationMethod} method to retrieve the ones
## common to all wells.
commonClassificationMethod(krasPlate)
```

`plateSummary` *Counts the number of positives and negatives in an experiment and produces estimates for the number of molecules.*

Description

Takes a collection of classified droplets, each corresponding to a well, and produces a list of positive/negative counts and estimates of how many molecules are in each well.

Usage

```
plateSummary(
  wells,
  ...,
  ch1Label = "Mt",
  ch2Label = "Wt",
  sortByLetter = FALSE
)

## S4 method for signature 'list'
plateSummary(wells, ch1Label = "Mt", ch2Label = "Wt", sortByLetter = FALSE)

## S4 method for signature 'ddpcrPlate'
plateSummary(
  wells,
  cMethod,
  ch1Label = "Mt",
  ch2Label = "Wt",
  sortByLetter = FALSE
)
```

Arguments

`wells`	Either a `ddpcrPlate` object or a list of data frames, each of which comprises droplet amplitudes and their corresponding classifications in a given well.
`...`	Other options depending on the type of `wells`.
`ch1Label`	The prefix to use for the channel 1 target. Defaults to "Mt".
`ch2Label`	The prefix to use for the channel 2 target. Defaults to "Wt".
`sortByLetter`	If `TRUE`, the resulting data frame is sorted by the letter in the well names first, e.g. "A02" comes before "B01". If `FALSE`, the result is sorted by the numeric component of the well names first, e.g. "B01" comes before "A02". Defaults to `FALSE`.
`cMethod`	The classification method to create a summary for.

Value

A data frame with droplet counts and molecules number estimates for each well.

Author(s)

Anthony Chiu, <anthony.chiu@cruk.manchester.ac.uk>

Examples

```
## Take a ddpcrPlate object and summarise its wells.
krasPlate <- ddpcrPlate(KRASdata)
plateSummary(krasPlate, cMethod="Cluster")
```

---

positiveCounts *Get a vector of droplet positive and negative counts.*

Description

Take a vector of classes and return a vector of positive and negative counts that is compatible with ddPCR analysis.

Usage

```
positiveCounts(cl)
```

Arguments

cl	A vector of classes that correspond to droplet amplitude data. The vector should only contain the values "PP", "PN", "NP" or "NN".

Value

A vector corresponding to "PP", "PN", "NP" and "NN".

Author(s)

Anthony Chiu, <anthony.chiu@cruk.manchester.ac.uk>

Examples

```
## Take a data frame and make it into the right format.
aWell <- KRASdata[["E03"]]
aWell$Cluster <- relabelClasses(aWell, classCol="Cluster")

## Count the number of droplets in each cluster.
positiveCounts(aWell$Cluster)
```

readCSVDataFrame *Read all given CSV files into a list.*

Description

Bio-Rad's QuantaSoft can export droplet amplitude data from multiple wells into CSV files for each well. This function can read these CSV files into a list. Note that empty wells will be ignored.

Usage

```
readCSVDataFrame(path, wellCol = FALSE, sortByLetter = FALSE)
```

Arguments

path	The path containing the CSV files (can be a combination of directories and individual CSV file paths). Each file will have a `Ch1.Amplitude`, `Ch2.Amplitude` and possibly classification columns, e.g. by default, QuantaSoft returns a `Cluster` column too.
wellCol	If `TRUE`, an additional column is added with the well name. This is useful if we need to merge all the data in the output list and we want to identify the original well of each droplet. Defaults to `FALSE`.
sortByLetter	If `TRUE`, the resulting list is sorted by the letter in the well names first, e.g. "A02" comes before "B01". If `FALSE`, the result is sorted by the numeric component of the well names first, e.g. "B01" comes before "A02". Defaults to `FALSE`.

Value

A list of data frames, each containing the data from a CSV file with the corresponding well name.

Author(s)

Anthony Chiu, <anthony.chiu@cruk.manchester.ac.uk>

Examples

```
## Read all of the droplet amplitudes from CSV files in a directory.
moreAmpsDir <- system.file("extdata", "more-amplitudes", package="twoddpcr")
someWells <- readCSVDataFrame(moreAmpsDir)

## We can read files from directories and files at the same time.
ampFile <- system.file("extdata", "amplitudes", "sample_B03_Amplitude.csv",
                       package="twoddpcr")
someWells <- readCSVDataFrame(c(moreAmpsDir, ampFile))

## If samples have been ordered by "A01", "A02", "A03", etc. instead of
## "A01", "B01", "C01", etc., we can set the sortByLetter flag to TRUE.
someWells <- readCSVDataFrame(moreAmpsDir, sortByLetter=TRUE)
```

```
## Setting wellCol to TRUE adds an extra column with the well name. If we
## bind the data frames together, we can track where the droplets came from.
someWells <- readCSVDataFrame(moreAmpsDir, wellCol=TRUE)
someWells <- do.call(rbind, someWells)
head(someWells)
tail(someWells)
```

`relabelClasses` *Re-label clusters.*

Description

An attempt to label clusters with "NN", "NP", "PN" or "PP" automatically. (This will not generalise to amplitudes in only a single channel.)

Usage

```
relabelClasses(droplets, classCol = "class", presentClasses = ddpcr$classes)
```

Arguments

`droplets`	A data frame of droplet amplitudes with a classification.
`classCol`	The column (name or number) from 'droplets' representing the class.
`presentClasses`	A vector of classes that we want to label. Must be a subset of c("NN", "NP", "PN", "PP") and must have the same number of classes as the number of unique classes in the class column.

Value

The classification column relabelled with "NN", "NP", "PN" and "PP".

Author(s)

Anthony Chiu, <anthony.chiu@cruk.manchester.ac.uk>

Examples

```
## Look at the "Cluster" column that was created by Bio-Rad's QuantaSoft.
aWell <- KRASdata[["E03"]]
str(aWell$Cluster)

## Relabel the classes to see the difference.
relabelled <- relabelClasses(aWell, classCol="Cluster")
str(relabelled)
levels(relabelled)

## We choose a sample with 3 clusters.
unique(KRASdata[["H04"]]$Cluster)
```

```
## We can check that there is no "PP" class, so specify the others only.
relabelled <- relabelClasses(KRASdata[["H04"]], classCol="Cluster",
                             presentClasses=c("NN", "PN", "NP"))
table(relabelled)
```

`removeDropletClasses` *Retrieve a data frame of droplet amplitudes with droplets of a given class removed.*

Description

By default, all droplets classified as "N/A" or "Rain" will be removed. Including these droplets is useful for visualisation purposes, but they could be a problem in some scenarios, e.g. if we wish to use the classification as a training data set.

Usage

```
removeDropletClasses(
  droplets,
  ...,
  classesToRemove = NULL,
  keepUnclassified = FALSE
)

## S4 method for signature 'data.frame'
removeDropletClasses(
  droplets,
  cMethod = "class",
  classesToRemove = NULL,
  keepUnclassified = FALSE
)

## S4 method for signature 'ddpcrWell'
removeDropletClasses(
  droplets,
  cMethod,
  classesToRemove = NULL,
  keepUnclassified = FALSE
)

## S4 method for signature 'ddpcrPlate'
removeDropletClasses(
  droplets,
  cMethod,
  classesToRemove = NULL,
  keepUnclassified = FALSE
)
```

Arguments

droplets	A ddpcrWell or ddpcrPlate object, or a data frame with columns Ch1.Amplitude, Ch2.Amplitude and a classification column.
...	Other parameters depending on the type of droplets.
classesToRemove	A vector of character strings corresponding to the classes that should be removed. Defaults to NULL, i.e. remove nothing.
keepUnclassified	A logical flag determining whether unclassified droplets (i.e. "Rain" or "N/A") should be kept. Defaults to FALSE, i.e. remove them.
cMethod	This is the name or column number corresponding to the classification in droplets. This column should only have entries in "NN", "PN", "NP, "PP", "Rain" and "N/A". Defaults to "class".

Value

If a ddpcrWell object is given, return a data frame corresponding to droplets with the given droplet classes removed. If a ddpcrPlate object is given, return a list of data frames instead.

Author(s)

Anthony Chiu, <anthony.chiu@cruk.manchester.ac.uk>

See Also

This function can remove "N/A" droplets from classifications produced by gridClassify.

Examples

```
## Take a data frame and transform it into the right format.
aWell <- KRASdata[["E03"]]
aWell$Cluster <- relabelClasses(aWell, classCol="Cluster")

## Add rain using the Mahalanobis distance.
aWell$ClusterMahRain <-
    mahalanobisRain(aWell, cMethod="Cluster", fullTable=FALSE)
table(aWell$ClusterMahRain)

## Suppose we want to use this for training. Remove the "Rain" droplets.
aWellCleaned <- removeDropletClasses(aWell, cMethod="ClusterMahRain")
table(aWellCleaned$ClusterMahRain)

## All of the above works with ddpcrWell objects.
aWell <- ddpcrWell(well=KRASdata[["E03"]])
aWell <- mahalanobisRain(aWell, cMethod="Cluster")
trainingData <- removeDropletClasses(aWell, cMethod="ClusterMahRain")
table(wellClassification(aWell, "ClusterMahRain"))
table(trainingData$ClusterMahRain)

## Likewise for ddpcrPlate objects we can create the training data.
```

```
krasPlate <- ddpcrPlate(wells=KRASdata[c("E03", "F03", "G03")])
krasPlate <- mahalanobisRain(krasPlate, cMethod="Cluster")
trainingData <- removeDropletClasses(krasPlate, cMethod="ClusterMahRain")
cl <- plateClassification(krasPlate, cMethod="ClusterMahRain")
cl <- unlist(cl)
table(cl)
td <- do.call(rbind, trainingData)
table(td$ClusterMahRain)

## We could also remove other droplet classes, such as the "PN" and "PP"
## clusters.
noPNPP <- removeDropletClasses(krasPlate, cMethod="ClusterMahRain",
                               classesToRemove=c("PN", "PP"))
td <- do.call(rbind, noPNPP)
table(td$ClusterMahRain)

## The same could be done, but with the "Rain" retained.
noPNPPWithRain <- removeDropletClasses(krasPlate, cMethod="ClusterMahRain",
                                       classesToRemove=c("PN", "PP"),
                                       keepUnclassified=TRUE)
td <- do.call(rbind, noPNPPWithRain)
table(td$ClusterMahRain)
```

`renormalisePlate` *Normalise all wells in a plate to the overall/average cluster positions.*

Description

Each well is scaled in the direction of the two axes independently. The algorithm works as follows:

1. Classify the entire plate using k-means clustering in order to roughly identify clusters.
2. For each channel and each well:
   (a) Remove the positive clusters in the other channel.
   (b) Re-run k-means clustering with $k = 2$ to obtain new cluster centres. If the centres are too close (default distance 2000), reject these new cluster centres and use the old centres.
   (c) Use the new centres for each well to rescale to the same scale as the average/overall scale.

Usage

```
renormalisePlate(
  plate,
  initialCentres = matrix(c(0, 0, 10000, 0, 0, 7000, 10000, 7000), ncol = 2, byrow =
    TRUE),
  minSeparation = 2000
)
```

Arguments

plate	A ddpcrPlate object.
initialCentres	A matrix or data frame of (rough estimates) of centres of each of the clusters in the combined data. This will be used for an initial run of k-means clustering.
minSeparation	The minimum distance required between two cluster centres in order for us to assume that k-means clustering found two distinct clusters. This is used when classifying droplets along a single channel and helps to reject classifications where there is only one cluster, e.g. when there are no mutants in a well. Defaults to 2000.

Value

A list of data frames with the rescaled amplitudes in both channels.

Author(s)

Anthony Chiu, <anthony.chiu@cruk.manchester.ac.uk>

Examples

```
## Normalise the KRAS data.
plate <- ddpcrPlate(wells=KRASdata)
normPlate <- renormalisePlate(plate)
```

---

sdRain *Add rain to a classification by using a chosen multiple of standard deviation.*

Description

Although we can use various algorithms to classify all droplets in a ddPCR experiment, there will be some variation between the classifications. We can perhaps have a relatively high confidence that droplets near the centres of clusters do indeed belong to that cluster, whereas we probably have a lower confidence in the classification of those further away, say, near the 'boundary' of two clusters. We may view these droplets (or a subset of them) as having an ambiguous class. This function allows us to only consider droplets classified within a certain distance of the means of each cluster and label the rest as "Rain".

Usage

```
sdRain(droplets, cMethod, errorLevel = 5, ...)

## S4 method for signature 'data.frame'
sdRain(droplets, cMethod, errorLevel = 5, fullTable = TRUE)

## S4 method for signature 'ddpcrWell'
```

```
sdRain(droplets, cMethod, errorLevel = 5)

## S4 method for signature 'ddpcrPlate'
sdRain(droplets, cMethod, errorLevel = 5)
```

Arguments

droplets	A ddpcrWell or ddpcrPlate object, or a droplet data frame including a classification column.
cMethod	The name or column number of the classification for which we want to add rain to.
errorLevel	How many multiples of standard deviation from the mean of each cluster to retain. Can be a list where each item corresponds to a class name and the multiple for that class. Can also be a numeric vector of length 1, which is equivalent to a list with all the same entries. Defaults to 5.
...	Other options depending on the type of droplets.
fullTable	If TRUE, returns a full data frame of droplets with an extra column of rainy data; if FALSE, simply returns a factor where each entry corresponds to an entry in the original classification column with added rain. Defaults to FALSE.

Value

If droplets is a data frame, return a data frame or factor (depending on fullTable) where droplets with ambiguous classifications are labelled as "Rain".

If droplets is a ddpcrWell object, return a ddpcrWell object with a rainy classification.

If droplets is a ddpcrPlate object, return a ddpcrPlate object with a rainy classifications.

Author(s)

Anthony Chiu, <anthony.chiu@cruk.manchester.ac.uk>

References

This approach was described in Jones, M., Williams, J., Gaertner, K., Phillips, R., Hurst, J., & Frater, J. (2014). Low copy target detection by Droplet Digital PCR through application of a novel open access bioinformatic pipeline, "definetherain." Journal of Virological Methods, 202(100), 46–53. http://doi.org/10.1016/j.jviromet.2014.02.020

Examples

```
## Compare the types of droplets in a single well for the "Cluster" class
## and then with rain.
aWell <- ddpcrWell(well=KRASdata[["E03"]])
aWell <- sdRain(aWell, cMethod="Cluster")
cl <- wellClassification(aWell)
table(cl$Cluster)
table(cl$ClusterSdRain)
```

```
## Compare the types of droplets in multiple wells for the "Cluster" class
## and then with rain.
krasPlate <- ddpcrPlate(wells=KRASdata[c("E03", "H03", "C04", "F04")])
krasPlate <- sdRain(krasPlate, cMethod="Cluster")
plateSummary(krasPlate, cMethod="Cluster")[, c(1:5)]
plateSummary(krasPlate, cMethod="ClusterSdRain")[, c(1:5)]

## The 'errorLevel' parameter can changed.
krasPlate <- sdRain(krasPlate, cMethod="Cluster", errorLevel=4)
plateSummary(krasPlate, cMethod="ClusterSdRain")[, c(1:5)]

## The 'errorLevel' parameter can also be changed for each cluster.
krasPlate <- sdRain(krasPlate, cMethod="Cluster",
                    errorLevel=list(NN=5, NP=5, PN=4, PP=3))
plateSummary(krasPlate, cMethod="ClusterSdRain")[, c(1:5)]
```

`setChannelNames` *Rename the channels.*

Description

We use the channel names "Ch1.Amplitude" and "Ch2.Amplitude" by default. This method allows us to change it at will.

Usage

```
setChannelNames(droplets, ch1 = "Ch1.Amplitude", ch2 = "Ch2.Amplitude")

## S4 method for signature 'data.frame'
setChannelNames(droplets, ch1 = "Ch1.Amplitude", ch2 = "Ch2.Amplitude")

## S4 method for signature 'list'
setChannelNames(droplets, ch1 = "Ch1.Amplitude", ch2 = "Ch2.Amplitude")
```

Arguments

`droplets`	A data frame or list of data frames. Each data frame should have at least two columns, where the first two columns should be vectors of doubles.
`ch1`	The channel 1 label.
`ch2`	The channel 2 label.

Value

The object `droplets` with the channels renamed.

Examples

```
## Ensure that a data frame has channels named "Ch1.Amplitude" and
## "Ch2.Amplitude".
aWell <- KRASdata[["E03"]]
aWell <- setChannelNames(aWell)
colnames(aWell)

## Perhaps we want to abbreviate the channel names.
aWell <- setChannelNames(aWell, "Ch1", "Ch2")
colnames(aWell)

## The same operator works for a list of data frames.
krasWells <- KRASdata
krasWells <- setChannelNames(krasWells, "Ch1", "Ch2")
lapply(krasWells, colnames)[1:3]
```

---

setDropletVolume *Set the droplet volume in microlitres (ul).*

Description

Change the volume of droplets used in the ddPCR experiment.

Usage

```
setDropletVolume(volume = 0.00085)
```

Arguments

volume	The new volume of each droplet. Defaults to 0.00085, which is the default given by Bio-Rad.

Value

Sets the internal droplet volume.

Author(s)

Anthony Chiu, <anthony.chiu@cruk.manchester.ac.uk>

Examples

```
## Change the droplet volume.
ddpcr$dropletVolume
setDropletVolume(0.00091)
ddpcr$dropletVolume
```

shinyVis *A Shiny environment.*

Description

Stores modes used the in Shiny UI and internal defaults.

Usage

```
shinyVis
```

Format

An object of class `environment` of length 3.

Value

An environment for storing Shiny variables.

shinyVisApp *Shiny app.*

Description

A shiny app for interactive ddPCR droplet classification.

Usage

```
shinyVisApp()
```

Value

A Shiny application

Author(s)

Anthony Chiu, <anthony.chiu@cruk.manchester.ac.uk>

shinyVisServer *Shiny visualisation server.*

Description

A Shiny server for interactive ddPCR droplet classification.

Usage

```
shinyVisServer(input, output, session)
```

Arguments

input	Shiny input list.
output	Shiny output list.
session	Shiny session.

Value

A Shiny server.

Author(s)

Anthony Chiu, <anthony.chiu@cruk.manchester.ac.uk>

shinyVisUI *Shiny visualisation UI.*

Description

A Shiny UI for interactive ddPCR droplet classification.

Usage

```
shinyVisUI()
```

Value

A Shiny UI.

Author(s)

Anthony Chiu, <anthony.chiu@cruk.manchester.ac.uk>

sortDataFrame *Sorts a data frame according to the well names.*

Description

Well names can be sorted according to the leading letter or the trailing digit.

Usage

```
sortDataFrame(df, sortByLetter = FALSE)
```

Arguments

df	A data frame with row names of the form A01, A02, B01, etc.
sortByLetter	If TRUE, the resulting list is sorted by the letter in the well names first, e.g. "A02" comes before "B01". If FALSE, the result is sorted by the numeric component of the well names first, e.g. "B01" comes before "A02". Defaults to FALSE.

Value

The data frame df with the rows sorted.

Author(s)

Anthony Chiu, <anthony.chiu@cruk.manchester.ac.uk>

Examples

```
## Sort a subset of the KRAScounts data frame.
df <- KRAScounts[, c("Ratio", "FracAbun")]
sortDataFrame(df=df, sortByLetter=FALSE)
sortDataFrame(df=df, sortByLetter=TRUE)
```

sortWells *Return given well names sorted.*

Description

Well names can be sorted according to the leading letter or the trailing digit.

Usage

```
sortWells(wellNames, sortByLetter)
```

Arguments

wellNames	The well names to sort.
sortByLetter	If TRUE, the resulting list is sorted by the letter in the well names first, e.g. "A02" comes before "B01". If FALSE, the result is sorted by the numeric component of the well names first, e.g. "B01" comes before "A02". Defaults to FALSE.

Value

The well names sorted.

Author(s)

Anthony Chiu, <anthony.chiu@cruk.manchester.ac.uk>

Examples

```
## Use the KRASdata dataset as an example.
plate <- ddpcrPlate(wells=KRASdata)
wellNames <- names(plate)

## Sample sorting.
sortWells(wellNames=wellNames, sortByLetter=FALSE)
sortWells(wellNames=wellNames, sortByLetter=TRUE)
```

---

textInputRow *Inline* textInputs.

Description

By default, textInput objects are added to the page below each other. This function allows for inline textInput fields.

Usage

```
textInputRow(inputId, label, value = "", size = NULL)
```

Arguments

inputId	The identifier for the textInput object.
label	The text label to display alongside.
value	The default value of the textInput.
size	The size (width) and maxlength.

Value

A div with the appropriate label and input field.

References

Adapted from: http://stackoverflow.com/a/21132918/1262569

`thresholdClassify` *Set thresholds to classify droplets.*

Description

Classify droplets as "NN", "NP", "PN" or "PP" depending on whether they lie above or below given thresholds. This is illustrated in the details and parameters for more detail.

Usage

```
thresholdClassify(droplets, ch1Threshold = 6500, ch2Threshold = 2900, ...)

## S4 method for signature 'data.frame'
thresholdClassify(
  droplets,
  ch1Threshold = 6500,
  ch2Threshold = 2900,
  fullTable = TRUE
)

## S4 method for signature 'ddpcrWell'
thresholdClassify(
  droplets,
  ch1Threshold = 6500,
  ch2Threshold = 2900,
  classMethodLabel = "thresholds"
)

## S4 method for signature 'ddpcrPlate'
thresholdClassify(
  droplets,
  ch1Threshold = 6500,
  ch2Threshold = 2900,
  classMethodLabel = "thresholds"
)
```

Arguments

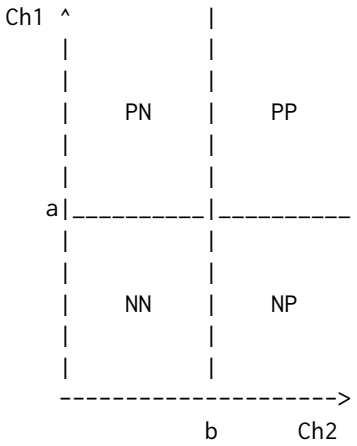
`droplets`	A `ddpcrWell` object or a data frame of droplet amplitudes with columns `Ch1.Amplitude` and `Ch2.Amplitude`.
`ch1Threshold`	The channel 1 upper bound for the NN and NP classes. Defaults to 8000
`ch2Threshold`	The channel 2 upper bound for the NN and PN classes. Defaults to 3000.
`...`	Other options depending on the type of `droplets`.

`fullTable` Whether to return a data frame including amplitude figures. If `TRUE`, a data frame with columns `Ch1.Amplitude`, `Ch1.Amplitude` and `class` is returned. If `FALSE`, a factor with levels in `ddpcr$classesRain` is returned, where each entry corresponds to each row in `droplets` (and `trainingData` is automatically set to `FALSE`). Defaults to `TRUE`.

`classMethodLabel`
A name (as a character string) of the classification method. Defaults to "thresholds".

Details

The `threshold` parameters correspond to those in the following diagram:



Specifically:

a: `ch1Threshold`,

b: `ch2Threshold`.

Value

If `droplets` is a data frame, return a data frame or factor (depending on the `trainingData` and `fullTable` parameters) with a classification for droplets in the chosen regions.

If `droplets` is a `ddpcrWell` object, return a `ddpcrWell` object with the appropriate classification.

If `droplets` is a `ddpcrPlate` object, return a `ddpcrPlate` object with the appropriate classification.

Author(s)

Anthony Chiu, <anthony.chiu@cruk.manchester.ac.uk>

See Also

This function is a special case of `gridClassify`.

Examples

```
## Use thresholds to set a classification for a data frame.
gCl <- thresholdClassify(KRASdata[["E03"]],
                         ch1Threshold=6789, ch2Threshold=3000)
str(gCl)

## The same works for ddpcrWell objects.
aWell <- ddpcrWell(well=KRASdata[["E03"]])
aWell <- thresholdClassify(aWell, ch1Threshold=6789, ch2Threshold=3000)
str(aWell)

## ddpcrPlate objects work in exactly the same way.
krasPlate <- ddpcrPlate(wells=KRASdata)
krasPlate <- thresholdClassify(krasPlate,
                               ch1Threshold=6789, ch2Threshold=3000)
lapply(plateClassification(krasPlate, withAmplitudes=TRUE), head, n=1)

## The default classification method (column name) is 'threshold'. It can be
## changed by providing a label.
krasPlate <- ddpcrPlate(wells=KRASdata)
krasPlate <- thresholdClassify(krasPlate,
                               ch1Threshold=6789, ch2Threshold=3000,
                               classMethodLabel="manual")
lapply(plateClassification(krasPlate, withAmplitudes=TRUE), head, n=1)
```

---

wellClassification *Retrieve a classification vector.*

Description

Retrieve the classification from a ddpcrWell object.

Usage

```
wellClassification(theObject, cMethod = NULL, withAmplitudes = FALSE)

## S4 method for signature 'ddpcrWell'
wellClassification(theObject, cMethod = NULL, withAmplitudes = FALSE)

wellClassification(theObject, cMethod) <- value

## S4 replacement method for signature 'ddpcrWell'
wellClassification(theObject, cMethod) <- value
```

Arguments

theObject	A ddpcrWell object.

cMethod	The names (or column numbers) of the classification to retrieve. If NULL, then all of the classifications are obtained. Defaults to NULL when retrieving. When setting a classification, this cannot be NULL.
withAmplitudes	Logical value. If TRUE, returns a data frame with the droplet amplitudes and corresponding classifications. If FALSE, returns the classification vector only. Defaults to FALSE.
value	A factor with the same length as the number of droplets in theObject, with levels in ddpcr$classesRain.

Value

A factor or data frame corresponding to the requested classification(s).

Author(s)

Anthony Chiu, <anthony.chiu@cruk.manchester.ac.uk>

See Also

wellClassificationMethod for the name of the classification method.

Examples

```
## Take some droplets with a given classification.
amplitudes <- KRASdata[["E03"]][, c("Ch1.Amplitude", "Ch2.Amplitude")]

## Create a ddpcrWell object with the amplitudes only.
aWell <- ddpcrWell(well=amplitudes)

## This has no classification yet.
head(wellClassification(aWell))

## We check the classification now, showing the amplitudes as well.
head(wellClassification(aWell, withAmplitudes=TRUE))

## Now set a sample classification.
wellClassification(aWell, cMethod="Sample") <-
  rep(c("NN", "NP", "PN", "PP"), numDroplets(aWell) / 4)
head(wellClassification(aWell, withAmplitudes=TRUE))
```

---

wellClassificationMethod

Retrieve the classification method.

Description

Retrieve the names of the classification methods for a ddpcrWell object.

Usage

```
wellClassificationMethod(theObject)

## S4 method for signature 'ddpcrWell'
wellClassificationMethod(theObject)

wellClassificationMethod(theObject, cMethod) <- value

## S4 replacement method for signature 'ddpcrWell'
wellClassificationMethod(theObject, cMethod) <- value
```

Arguments

theObject	A ddpcrWell object.
cMethod	If modifying the classification methods, this should be a vector of existing classification names or numbers.
value	A character vector (of the same length as cMethod) giving a new names for the chosen classification methods.

Value

The classification method names.

Author(s)

Anthony Chiu, <anthony.chiu@cruk.manchester.ac.uk>

See Also

wellClassification for the classification of the droplets.

Examples

```
## Create a ddpcrWell object with some data and classification.
aWell <- ddpcrWell(well=KRASdata[["E03"]])

## Retrieve the classification method names.
head(wellClassificationMethod(aWell))

## Set a classification method name to something new.
wellClassificationMethod(aWell, cMethod="Cluster") <- "QuantaSoft"

## We check the classification now, showing the amplitudes as well.
wellClassificationMethod(aWell)
```

whiteTheme *Use a theme with a white background and grey lines.*

Description

The default ggplot theme has a grey background. This layer can be added to change the aesthetics.

Usage

```
whiteTheme(legendPosition = "right", legendJustification = "center")
```

Arguments

legendPosition	Where to position the legend. This can be "none", "left", "right", "bottom", "top" or a numeric vector of length two. See the documentation for theme.
legendJustification	The justification to use for the legend positioning. This should be "center" or a two-element vector. Defaults to "center".

Value

A ggplot theme layer.

Author(s)

Anthony Chiu, <anthony.chiu@cruk.manchester.ac.uk>

See Also

This function uses theme.

Examples

```
## Get a data frame in the right format.
aWell <- KRASdata[["E03"]]
aWell$Cluster <- relabelClasses(aWell, classCol="Cluster")

## Plot it with the white theme.
ggplot2::ggplot(aWell, ggplot2::aes(x=Ch2.Amplitude, y=Ch1.Amplitude)) +
  ggplot2::geom_point(ggplot2::aes(colour=Cluster)) +
  whiteTheme()
```

`wildTypeCopiesSummary` *Get the wild type copies per 20ul of a data frame in the context of the basic counts.*

Description

Returns a data frame with the basic figures and the number of wild type copies per 20ul.

Usage

```
wildTypeCopiesSummary(df)
```

Arguments

`df`	A data frame generated by `fullCountsSummary`.

Value

A data frame with the basic columns and wild type copies per 20ul.

Author(s)

Anthony Chiu, <anthony.chiu@cruk.manchester.ac.uk>

Examples

```
df <- fullCountsSummary(KRAScountsQS)
wildTypeCopiesSummary(df)
```

Index

∗ **datasets**
ddpcr, 26
KRAScounts, 58
KRASdata, 59
shinyVis, 80
.classifyDfOnChannel, 4
.classifyOnChannel, 5
.classwiseMahalanobisRain, 6
.cov, 7
.ddpcrPlate *(*ddpcrPlate-class*)*, 26
.ddpcrWell *(*ddpcrWell-class*)*, 27
.dependentCols, 7
.essentialDependentCols, 8
.extractWellNames, 8
.getAllSummary, 9
.getChannelCentres, 9
.getClassificationData, 10
.getMutCopies, 11
.getWellNames, 11
.getWtCopies, 12
.isTwoDimDataFrame, 12
.isWideForm, 13
.mahDist, 13
.matrixInverse, 14
.numberOfWells, 14
.renormaliseByChannel, 15
.renormaliseWell, 16
.roundIt, 16
.slice, 17
.totalCopies, 17

allPlot *(*facetPlot*)*, 38
amplitudes, 18, *66*
amplitudes,ddpcrPlate-method *(*amplitudes*)*, 18
amplitudes,ddpcrWell-method *(*amplitudes*)*, 18

basicsSummary, 19

castSummary, 20
classCov, 21
classMeans, 22
classStats, *13*, 23
clusterCenters *(*clusterCentres*)*, 24
clusterCentres, 24
clusterCentres,ddpcrPlate-method *(*clusterCentres*)*, 24
clusterCentres,ddpcrWell-method *(*clusterCentres*)*, 24
combinedCenters *(*clusterCentres*)*, 24
combinedCentres *(*clusterCentres*)*, 24
combinedCentres,ddpcrPlate-method *(*clusterCentres*)*, 24
combinedPlateCenters *(*clusterCentres*)*, 24
combinedPlateCentres *(*clusterCentres*)*, 24
commonClassificationMethod *(*plateClassificationMethod*)*, 67
commonClassificationMethod,ddpcrPlate-method *(*plateClassificationMethod*)*, 67
copiesSummary, 25

ddpcr, 26
ddpcrPlate, *8*, *18*, *24*, *29*, *31*, *34*, *38*, *41*, *45*, *54*, *56*, *62*, *66*, *68*, *69*, *74*, *77*
ddpcrPlate *(*ddpcrPlate-class*)*, 26
ddpcrPlate,character-method *(*ddpcrPlate-class*)*, 26
ddpcrPlate,ddpcrPlate-method *(*ddpcrPlate-class*)*, 26
ddpcrPlate,list-method *(*ddpcrPlate-class*)*, 26
ddpcrPlate,missing-method *(*ddpcrPlate-class*)*, 26
ddpcrPlate-class, 26
ddpcrWell, *18*, *24*, *27*, *29*, *31*, *34*, *41*, *45*, *48*, *53*, *54*, *56*, *62*, *74*, *77*, *84*, *86–88*
ddpcrWell *(*ddpcrWell-class*)*, 27

ddpcrWell,character-method
(ddpcrWell-class), 27
ddpcrWell,data.frame-method
(ddpcrWell-class), 27
ddpcrWell,ddpcrWell-method
(ddpcrWell-class), 27
ddpcrWell,missing-method
(ddpcrWell-class), 27
ddpcrWell-class, 27
densityPlot (heatPlot), 51
drawBlank, 29
dropletPlot, *29*, 29, *46*
dropletPlot,data.frame-method
(dropletPlot), 29
dropletPlot,ddpcrPlate-method
(dropletPlot), 29
dropletPlot,ddpcrWell-method
(dropletPlot), 29

elementType,SimpleList-method, 32
exportTable, 33
exportTable,data.frame-method
(exportTable), 33
exportTable,ddpcrPlate-method
(exportTable), 33
exportTable,ddpcrWell-method
(exportTable), 33
exportZip (exportTable), 33
exportZip,ddpcrPlate-method
(exportTable), 33
extractPlateName, 36
extractWellNames, 37

facet_grid, *39*
facet_wrap, *39*
facetPlot, 38
facetPlot,data.frame-method
(facetPlot), 38
facetPlot,ddpcrPlate-method
(facetPlot), 38
flatPlot, 40
flatPlot,data.frame-method (flatPlot),
40
flatPlot,ddpcrPlate-method (flatPlot),
40
flatPlot,ddpcrWell-method (flatPlot), 40
fullCopiesSummary, 42
fullCountsSummary, *9*, *11*, *12*, *19*, *25*, *42*, 43,
61, *90*

getCutOff, 44
ggplot, *31*, *39*, *45*, *46*, *52*, *89*
ggplot.multiwell (ggplot.well), 45
ggplot.plate, *29*, *31*
ggplot.plate (ggplot.well), 45
ggplot.plate,ddpcrPlate-method
(ggplot.well), 45
ggplot.well, *29*, *31*, 45
ggplot.well,ddpcrWell-method
(ggplot.well), 45
ggplot.wells (ggplot.well), 45
gridClassify, 47, *74*, *85*
gridClassify,data.frame-method
(gridClassify), 47
gridClassify,ddpcrPlate-method
(gridClassify), 47
gridClassify,ddpcrWell-method
(gridClassify), 47

heatPlot, *39*, 51
heatPlot,data.frame-method (heatPlot),
51
heatPlot,ddpcrPlate-method (heatPlot),
51
heatPlot,ddpcrWell-method (heatPlot), 51

isEmpty,ddpcrPlate-method
(isEmpty,ddpcrWell-method), 53
isEmpty,ddpcrWell-method, 53

kmeans, *55*
kmeansClassify, 54
kmeansClassify,data.frame-method
(kmeansClassify), 54
kmeansClassify,ddpcrPlate-method
(kmeansClassify), 54
kmeansClassify,ddpcrWell-method
(kmeansClassify), 54
knn, *57*
knnClassify, *55*, 56
knnClassify,data.frame-method
(knnClassify), 56
knnClassify,ddpcrPlate-method
(knnClassify), 56
knnClassify,ddpcrWell-method
(knnClassify), 56
KRAScounts, 58
KRAScountsQS (KRAScounts), 58
KRAScountsWellCol (KRAScounts), 58

KRASdata, *58*, *59*, 59

mahalanobisRain, 60
mahalanobisRain,data.frame-method (mahalanobisRain), 60
mahalanobisRain,ddpcrPlate-method (mahalanobisRain), 60
mahalanobisRain,ddpcrWell-method (mahalanobisRain), 60
multivariateNormalRain (mahalanobisRain), 60
multivariateRain (mahalanobisRain), 60
mutantCopiesSummary, 61
mvNormalRain (mahalanobisRain), 60
mvnRain (mahalanobisRain), 60

numberDroplets (numDroplets), 62
numberDrops (numDroplets), 62
numberOfDroplets (numDroplets), 62
numberOfDrops (numDroplets), 62
numDroplets, 62
numDroplets,ddpcrPlate-method (numDroplets), 62
numDroplets,ddpcrWell-method (numDroplets), 62
numDrops (numDroplets), 62
numericInput, *63*
numericInputRow, 63
numOfDroplets (numDroplets), 62

parseClusterCounts, 64
plateCenters (clusterCentres), 24
plateCentres (clusterCentres), 24
plateClassification, *55*, *57*, 65, *68*
plateClassification,ddpcrPlate-method (plateClassification), 65
plateClassification<- (plateClassification), 65
plateClassification<-,ddpcrPlate,character,factor-method (plateClassification), 65
plateClassification<-,ddpcrPlate,character,list-method (plateClassification), 65
plateClassificationMethod, *55*, *57*, 67
plateClassificationMethod,ddpcrPlate-method (plateClassificationMethod), 67
plateClassificationMethod<- (plateClassificationMethod), 67
plateClassificationMethod<-,ddpcrPlate-method (plateClassificationMethod), 67
plateClassificationName (plateClassificationMethod), 67
plateClassMethod (plateClassificationMethod), 67
plateClassName (plateClassificationMethod), 67
plateSummary, 69
plateSummary,ddpcrPlate-method (plateSummary), 69
plateSummary,list-method (plateSummary), 69
plotAll (facetPlot), 38
positiveCounts, 70

readCSVDataFrame, *34*, 71
relabelClasses, 72
removeDropletClasses, *50*, 73
removeDropletClasses,data.frame-method (removeDropletClasses), 73
removeDropletClasses,ddpcrPlate-method (removeDropletClasses), 73
removeDropletClasses,ddpcrWell-method (removeDropletClasses), 73
renormalisePlate, 75

sdRain, 76
sdRain,data.frame-method (sdRain), 76
sdRain,ddpcrPlate-method (sdRain), 76
sdRain,ddpcrWell-method (sdRain), 76
setChannelNames, 78
setChannelNames,data.frame-method (setChannelNames), 78
setChannelNames,list-method (setChannelNames), 78
setDropletVol (setDropletVolume), 79
setDropletVolume, 79
setDropVol (setDropletVolume), 79
setDropVolume (setDropletVolume), 79
shinyVis, 80
shinyVisApp, 80
shinyVisServer, 81
shinyVisUI, 81
show,ddpcrPlate-method (ddpcrPlate-class), 26
show,ddpcrWell-method (ddpcrWell-class), 27
sortDataFrame, 82
sortWells, 82

textInput, *83*
textInputRow, 83
theme, *89*
thresholdClassify, *50*, 84
thresholdClassify,data.frame-method
(thresholdClassify), 84
thresholdClassify,ddpcrPlate-method
(thresholdClassify), 84
thresholdClassify,ddpcrWell-method
(thresholdClassify), 84
twoddpcr (twoddpcr-package), 4
twoddpcr-package, 4

wellCenters (clusterCentres), 24
wellCentres (clusterCentres), 24
wellClassification, *18*, *55*, *57*, 86, *88*
wellClassification,ddpcrWell-method
(wellClassification), 86
wellClassification<-
(wellClassification), 86
wellClassification<-,ddpcrWell-method
(wellClassification), 86
wellClassificationMethod, *87*, 87
wellClassificationMethod,ddpcrWell-method
(wellClassificationMethod), 87
wellClassificationMethod<-
(wellClassificationMethod), 87
wellClassificationMethod<-,ddpcrWell-method
(wellClassificationMethod), 87
whiteTheme, 89
wildTypeCopiesSummary, 90