# cellHTS

October 25, 2011

---

`Bscore` *B score normalization*

---

## Description

Correction of plate and spatial effects of the data `xraw` of a `cellHTS` object using the B score method. Using this method, a two-way median polish is fitted in a per-plate basis to account for row and column effects. Optionally, the obtained residuals within each plate can be further divided by their median absolute deviations to standardize for plate-to-plate variability. Optionally, a transformation to $z$-scores can be performed.

## Usage

```
Bscore(x, what="xraw", adjustPlateMedian = TRUE, scale = TRUE, save.model = FALS
```

## Arguments

| | |
|---|---|
| `x` | a `cellHTS` object that has already been configured. See details. |
| `what` | a character indicating the slot of `x` to consider: `"xraw"` (default) or `"xnorm"`. |
| `adjustPlateMedian` | a logical value indicating whether the estimated average of each plate should also be substracted to the raw intensity values. |
| `scale` | a logical value indicating if the per-plate model residuals should be further scaled by their variance. See details. |
| `save.model` | a logical value specifying whether the per-plate models should be saved, and given as output. See details. |

## Details

The normalization is performed in a per-plate fashion using the B score method. This function can be called direclty, or indirectly, using `normalizePlates` or `summarizeChannels`. In the B score method, the residual $r_{ijp}$ of the measurement for row $i$ and column $j$ on the $p$-th plate is obtained by fitting a two-way median polish, in order to account for both row and column effects within the plate:

$$r_{ijp} = y_{ijp} - \hat{y}_{ijp} = y_{ijp} - (\hat{\mu}_p + \hat{R}_{ip} + \hat{C}_{jp})$$

$y_{ijp}$ is the measurement value in row *i* and column *j* of plate *p* (taken from `x$xraw`), and $\hat{y}_{ipj}$ is the corresponding fitted value. This is defined as the sum between the estimated average of the plate ($\hat{\mu}_p$), the estimated systematic offset for row *i* ($\hat{R}_{ip}$), and the systematic offset for column *j* ($\hat{C}_{jp}$).

If `scale=TRUE`, for each plate *p*, each of the obtained residual values $r_{ijp}$'s are divided by the median absolute deviation of the residuals in plate *p* ($MAD_p$), giving the B score value:

$$B_{score_{ijp}} = \frac{r_{ijp}}{MAD_p}$$

If `adjustPlateMedian` is set to `FALSE`, the estimated overall plate average ($\hat{\mu}_p$) is not removed from the intensity values $y_{ijp}$'s.

If `save.model=TRUE`, the models residuals ($r_{ijp}$'s), row and column offsets and overall offsets are stored in the slots `residuals`, `rowcol.effects` and `overall.effects` of the `cellHTS` object `x`.

## Value

An object of class `cellHTS`, which is a copy of the argument `x`, plus an additional slot `xnorm` containing the normalized data. This is an array of the same dimensions as `xraw`. Furthermore, if `save.model=TRUE`, the slots `residuals`, `rowcol.effects`, and `overall.effects` (only if `adjustPlateMedian` was also set to `TRUE`) are added to `x`. The latter slots are arrays with the same dimension as `x$xraw`, except the `overall.effects`, which have dimensions `1 x nr Plates x nr Replicates x nr Channels`.

Moreover, the processing status of the `cellHTS` object is updated in the slot `state` to `x$state["normalized"]=`

## Author(s)

Ligia Braz `<ligia@ebi.ac.uk>`

## References

Brideau, C., Gunter, B., Pikounis, B. and Liaw, A. (2003) Improved statistical methods for hit selection in high-throughput screening, *J. Biomol. Screen* **8**, 634–647.

Malo, N., Hanley, J.A., Cerquozzi, S., Pelletier, J. and Nadon, R. (2006) Statistical practice in high-throughput screening data analysis, *Nature Biotechn* **24**(2), 167–175.

## See Also

`medpolish`, `plotSpatialEffects`, `normalizePlates`, `summarizeChannels`

## Examples

```
data(KcViabSmall)
x <- KcViabSmall
x <- Bscore(x, save.model = TRUE)
## identical result, but calling Bscore function from "normalizePlates"
xopt <- normalizePlates(x, normalizationMethod="Bscore", save.model = TRUE)
all(x$xnorm==xopt$xnorm, na.rm=TRUE)
```

ROC *Creates an object of class "ROC" which can be plotted as a ROC curve*

## Description

The function ROC construct an object of S3 class ROC, which represents a receiver-operator-characteristic curve, from the data of the annotated positive and negative controls in a scored cellHTS object.

## Usage

```
ROC(x, positives, negatives)
## S3 method for class 'ROC'
plot(x, col="darkblue", type="l", main = "ROC curve", ...)
## S3 method for class 'ROC'
lines(x, ...)
```

## Arguments

| | |
|---|---|
| x | a cellHTS object that has already been scored (see details). |
| positives | a list or vector of regular expressions specifying the name of the positive controls. See the details for the argument posControls of writeReport function. |
| negatives | a vector of regular expressions specifying the name of the negative controls. See the details for the argument negControls of writeReport function. |
| col | the graphical parameter for color; see par for details. |
| type | the graphical parameter giving the type of plot desired; see par for details. |
| main | the graphical parameter giving the desired title of plot; see par for details. |
| ... | other graphical parameters as in par may be also passed as arguments. |

## Details

The cellHTS object x must contain a slot called score, and selection proceeds from large to small values of this score. Furthermore, x is expected to contain positive and negative controls annotated in the slot wellAnno with the values of the arguments positives and negatives, respectively. If the assay is a two-way experiment, positives should be a list with components act and inh, specifying the name of the activators, and inhibitors, respectively. In this case, the ROC cureve is constructed based on the absolute values of x$score.

## Value

An S3 object of class ROC. There are methods plot.ROC and lines.ROC.

## Author(s)

Ligia P. Bras <ligia@ebi.ac.uk>

## Examples

```
data(KcViabSmall)
## Not run:
x <- normalizePlates(KcViabSmall, normalizationMethod="median", zscore="-")
x <- summarizeReplicates(x)
y <- ROC(x)
plot(y)
lines(y)

## End(Not run)
```

---

`annotate` *Annotates the gene IDs of a given cellHTS object*

---

## Description

Annotate the gene IDs of a given `cellHTS` object.

## Usage

```
annotate(x, ...)
## S3 method for class 'cellHTS'
annotate(x, geneIDFile, path, ...)
```

## Arguments

| | |
|---|---|
| `x` | a cellHTS object. |
| `geneIDFile` | the name of the file with the gene IDs (see details). This argument is just passed on to the `read.table` function, so any of the valid argument types for `read.table` are valid here, too. Must contain one row for each well and each plate. |
| `path` | a character of length 1 indicating the path in which to find the gene annotation file. By default, it can extract the path from `geneIDFile`. |
| `...` | additional parameters - ignored. |

## Details

- geneIDFileThis file is expected to be a tab-delimited file with at least three columns, and column names `Plate`, `Well` and `GeneID`. The contents of `Plate` are expected to be integer. Further columns are allowed.

## Value

An S3 object of class `cellHTS`, which extends the argument `x` by the following element:

| | |
|---|---|
| `geneAnno` | a `data.frame` containing what was read from input file `geneIDFile`. The number of rows is equal to the product between the number of wells in each plate and the number of plates. |

Moreover, the processing status of the `cellHTS` object is updated in the slot `state` to `x$state["annotated"]= TRUE`. There are methods `print.cellHTS`, `configure.cellHTS` and `annotate.cellHTS`.

### Author(s)

Wolfgang Huber <huber@ebi.ac.uk>, Ligia Braz <ligia@ebi.ac.uk>

### References

..

### See Also

readPlateData, configure

### Examples

```
## Not run:
    datadir <- system.file("KcViabSmall", package = "cellHTS")
    x <- readPlateData("Platelist.txt", "KcViabSmall", path=datadir)
    x <- configure(x, "Plateconf.txt", "Screenlog.txt", "Description.txt", path=datadir)
    x <- annotate(x, "GeneIDs_Dm_HFAsubset_1.1.txt", path=datadir)

## End(Not run)
```

---

## bdgpbiomart *Dataset with annotation of CG identifiers*

---

### Description

See the complete vignette *End-to-end analysis of cell-based screens: from raw intensity readings to the annotated hit list*, Section *Using biomaRt to annotate the target genes online* for details. The annotations were obtained on 8 September 2006.

### Usage

```
data(bdgpbiomart)
```

### Format

Dataframe with 21888 rows and 11 columns Plate, Well, HFAid, GeneID, chr_name, chrom_start, chrom_end, description, flybase_name, go_id, go_description.

### Source

BioMart webinterface to Ensembl 37.

### Examples

```
data(bdgpbiomart)
```

`configure` *Configures the plates and plate result files*

## Description

Annotate the plates and the plate result files of a given `cellHTS` object.

## Usage

```
configure(x, ...)
## S3 method for class 'cellHTS'
configure(x, confFile, logFile, descripFile, path, ...)
```

## Arguments

| | |
|---|---|
| `x` | a `cellHTS` object. |
| `confFile` | the name of the configuration file (see details). This argument is just passed on to the `read.table` function, so any of the valid argument types for `read.table` are valid here, too. Must contain one row for each well and each batch. |
| `logFile` | optional; the name of the screen log file (see details). This argument is just passed on to the `read.table` function, so any of the valid argument types for `read.table` are valid here, too. |
| `descripFile` | the name of the screen description file (see details). This argument is just passed on to the `readLines` function, so any of the valid argument types for `readLines` are valid here, too. |
| `path` | optional; a character of length one indicating the path in which to find the configuration files. Useful when the files are locate in the same directory, but should be omitted otherwise. |
| `...` | additional parameters - ignored. |

## Details

- confFileThis file is expected to be a tab-delimited file with at least three columns, and column names `Batch`, `Well` and `Content`. The contents of `Batch` are expected to be integer.
- logFileIf given as an argument, it is expected to be a tab-delimited file with at least three columns, and column names `Filename`, `Well`, and `Flag`. Further columns are allowed.
- descripFileThis file is the screen description file with general information about the screen.

Data from wells that are annotated as *empty* are ignored and are set to NA in `x` in slot `xraw`.

## Value

An S3 object of class `cellHTS`, which extends the argument `x` by the following elements:

| | |
|---|---|
| `plateConf` | a data.frame containing what was read from input file `confFile`. The number of rows is equal to the product between the number of wells in each plate and the number of batches. |
| `screenLog` | a data.frame containing what was read from input file `logFile`. |
| `screenDesc` | object of class `character` containing what was read from input file `descripFile`. |

Moreover, the processing status of the `cellHTS` object is updated in the slot `state` to `state["configured"]=TRU`

`wellAnno` object of class `factor` of length number of plates x number of wells per plate, with possible levels: *empty*, *other*, *neg*, *sample*, and *pos*, indicative of the contents of the wells. Other levels may be employed for the positive and negative controls, besides *pos* and *neg*.

There are methods `print.cellHTS`, `configure.cellHTS` and `annotate.cellHTS`.

## Author(s)

Wolfgang Huber <huber@ebi.ac.uk>, Ligia Braz <ligia@ebi.ac.uk>

## References

..

## See Also

`readPlateData`

## Examples

```
## Not run:
    datadir <- system.file("KcViabSmall", package = "cellHTS")
    x <- readPlateData("Platelist.txt", "KcViabSmall", path=datadir)
    x <- configure(x, "Plateconf.txt", "Screenlog.txt", "Description.txt", path=datadir)

## End(Not run)
```

---

# KcViab *A sample cellHTS object - D. melanogaster genome-wide RNAi screen*

---

## Description

Archived `cellHTS` object from a genome-wide RNAi screen of cell viability in *Drosophila* Kc167 cells

## Usage

```
##cellHTS object, see examples for details
```

## Format

`cellHTS` object

## References

Boutros, M., Kiger, A.A., Armknecht,S., Kerr,K., Hild,M., Koch,B., Haas, S.A., Heidelberg Fly Array Consortium, Paro,R. and Perrimon, N. (2004) Genome-wide RNAi analysis of growth and viability in Drosophila cells, *Science* **303**:832–5.

## Examples

```
data(KcViab)
```

## KcViabSmall *A sample cellHTS object - D. melanogaster genome-wide RNAi screen*

### Description

Archived `cellHTS` object corresponding to the first three 384-well plates of a genome-wide RNAi screen of cell viability in *Drosophila* Kc167 cells

### Usage

```
##cellHTS object, see examples for details
```

### Format

`cellHTS` object

### References

Boutros, M., Kiger, A.A., Armknecht,S., Kerr,K., Hild,M., Koch,B., Haas, S.A., Heidelberg Fly Array Consortium, Paro,R. and Perrimon, N. (2004) Genome-wide RNAi analysis of growth and viability in Drosophila cells, *Science* **303**:832–5.

### Examples

```
data(KcViabSmall)
```

## getEnVisionRawData *Read a plate file obtain from EnVision Plate Reader*

### Description

Import functions to read a plate file obtained from EnVision Plate Reader. These functions should be set as the import function of `readPlateData` through the argument `importFun` when reading plate result files obtained from EnVision plate reader.

### Usage

```
getEnVisionRawData(f)
getEnVisionCrosstalkCorrectedData(f)
```

### Arguments

| | |
|---|---|
| `f` | the name of the result plate file to read. |

### Details

These functions should not be called directly. Instead, they should be set as the import function of `readPlateData` through the argument `importFun` when reading plate result files obtained from an EnVision plate reader.

## Value

These functions return a list with two components. The first component should be a 'data.frame' with the following slots: `well` (a character vector with the well identifier in the plate) and `val` (the intensity values measured at each well). The second component of this list should be a character vector containing a copy of the imported input data file (such as the output of `readLines`). It should be suitable to be used as input for `writeLines`, since it will be used to reproduce the intensity files that are linked in the HTML quality reports generated by `writeReport`.

## Author(s)

Ligia Bras <ligia@ebi.ac.uk>

## See Also

`readPlateData`

## Examples

```
plateFile <- system.file("EnVisionExample/XXX_1500.csv", package = "cellHTS")
onePlate <- getEnVisionRawData(plateFile)

## to get the cross talk corrected data:
onePlate2 <- getEnVisionCrosstalkCorrectedData(plateFile)
```

---

# getLibraryPlate *384-well plate assay format to a 96-well plate library format*

---

## Description

Given a `cellHTS` object with data from an assay conducted in 384-well plate format, resulting from the combination of four consecutive 96-well plates of a reagent library, this function gives the plate identifiers for the 96-well plates.

## Usage

```
getLibraryPlate(x)
```

## Arguments

| | |
|---|---|
| `x` | a `cellHTS` object. |

## Details

The `cellHTS` object `x` contains data from a screening experiment where every set of four consecutive 96-well plates was combined into a 384-well plate. Therefore, the only available plate identifiers are for the assay plate format (384-well plates). The way the four 96-well plates are transferred to a 384-well plate during an experiment is as follows: the robot stars by transferring the samples from the first 96-well plate into the first quadrant of the 384-well plate, and so on.

### Value

An S3 object of class `cellHTS`, which extends the argument `x` by the following element:

`libPlate` a vector of length equal to the total number of wells of all the 384-well plates, containing a number that identifies the 96-well plate. It ranges from 1 to four times the total number of 384-well plates.

### Author(s)

Ligia Braz <ligia@ebi.ac.uk>

### Examples

```
data(KcViabSmall)
x <- getLibraryPlate(KcViabSmall)
table(x$libPlate)
```

---

## getMatrix *Create a matrix with replicate data in columns*

---

### Description

Given an array of raw or normalized intensities (`xraw` or `xnorm`) of a `cellHTS` object, creates a matrix with the data from the chosen channel.

### Usage

```
getMatrix(y, channel=1, na.rm=FALSE)
```

### Arguments

| | |
|---|---|
| `y` | an array with four dimensions, such as the slot `xraw` or `xnorm` of a `cellHTS` object. |
| `channel` | a numeric value corresponding to the selected channel of `y`. By default, the first channel (that is, `y[,,1]` is considered). |
| `na.rm` | Logical, indicated if the missing values should be omitted. |

### Details

Given as input an array `y` (e.g., the slot `xraw`, or `xnorm` of a `cellHTS` object) with dimensions `nr wells x nr plates x nr replicates x nr channels`, this function creates a matrix with the data for the chosen `channel`. Each replicate corresponds to a column of the output matrix. If `na.rm` is set to `TRUE`, only the positions with available values for all the replicates are given in the output matrix.

### Value

A matrix with the same number of columns as the number of replicates (third dimension of `y`). If `na.rm=FALSE` (the default), the number of rows of the output matrix is identical to the product between the first two dimensions of `y` (`nr wells x nr plates`). If `na.rm=TRUE`, only the rows with no missing entries in all the replicates (columns) are given.

### Author(s)

Ligia Bras <ligia@ebi.ac.uk>

### Examples

```
data(KcViabSmall)
y <- getMatrix(KcViabSmall$xraw)
```

---

## imageScreen *Experiment-wide quality control plot of a cellHTS object*

---

### Description

Experiment-wide quality control plot of a scored cellHTS object.

### Usage

```
imageScreen(x, ar = 3/5, zrange, map = FALSE, anno)
```

### Arguments

| | |
|---|---|
| x | a cellHTS object that has already been scored (i.e. containing the slot score). |
| ar | the desired aspect ration for the image plot (i.e. number of columns per number of rows) |
| zrange | the range of values to be mapped into the color scale. If missing, zrange will be set to the range of x$score. |
| map | a logical value that determines whether an image map should be created using tooltips to indicate the annotation at each position. It only makes sense to set it to TRUE when the function is called from writeReport, so the default is FALSE. |
| anno | optional input giving the annotation information for the mapping. It should be a vector of the same size as x$score. See details. |

### Details

This function creates an image plot that gives an overview of the whole set of score values from the cellHTS object x, x$score. When the annotation mapping is performed, by default, anno is set to:

1. The content of x$geneAnno$GeneSymbol (or x$geneAnno$GeneID, if the former is not available), if x is annotated;
2. The position within the plate, if x is not annotated yet.

### Author(s)

Ligia Braz <ligia@ebi.ac.uk>

### See Also

normalizePlates, summarizeChannels, summarizeReplicates, writeReport

## Examples

```
data(KcViabSmall)
x <- KcViabSmall
x <- normalizePlates(x, normalizationMethod="median", zscore="-")
x <- summarizeReplicates(x)
imageScreen(x, zrange=c(-5,5))
```

---

`normalizeChannels` *Normalization of dual-channel data and data transformation*

---

## Description

Normalizes and/or transforms dual-channel data `xraw` of a given `cellHTS` object by applying the function defined in `fun`. The default is to take the ratio between the second and first channels ($\frac{r_2}{r_1}$). Correction of plate-to-plate variations may also be performed.

## Usage

```
normalizeChannels(x, fun = function(r1,r2) r2/r1, log = FALSE,
                  adjustPlates, zscore, posControls, negControls, ...)
```

## Arguments

| | |
|---|---|
| `x` | a cellHTS object that has already been configured. See details. |
| `fun` | a function defined by the user to relate the signal in the two channels `r1` and `r2`. `fun` takes two numeric vectors and returns a numeric vector of the same length. The default is to take the ratio between the second and first channels. |
| `log` | a logical value indicating whether the result obtained after applying `fun` should be `log2` transformed. The default is `log = FALSE`, and the data is not `log2` transformed. |
| `adjustPlates` | character string indicating the correction method to apply to adjust for plate-to-plate variations (and eventually well-to-well variations), after applying `fun` and eventually log transforming the values. Allowed values are `"median"`, `"mean"`, `"shorth"`, `"POC"`, `"NPI"`, `"negatives"` and `Bscore`. If `adjustPlates` is missing (the default), no plate-wise correction will be performed. See details. |
| `zscore` | indicates if the $z$-scores should be determined after normalization and transformation. If missing (default), the data will not be scored. Otherwise, it should be a character string, either "+" or "-", specifying the sign to use for the calculated $z$-scores. See details. |
| `posControls` | a vector of regular expressions giving the name of the positive control(s). See details. |
| `negControls` | a vector of regular expressions giving the name of the negative control(s). See details. |
| `...` | Further arguments that get passed on to the function implementing the normalization method chosen by `adjustPlates`. Currently, this is only used for `Bscore`. |

## Details

For each plate and replicate of a two-color experiment, the function defined in `fun` is applied to relate the intensity values in the two channels of the `cellHTS` object. The default is to calculate the ratio between the second and the first channels, but other options can be defined.

If `log = TRUE`, the data obtained after applying `fun` is `log2` transformed. The default is `log = FALSE`.

If `adjustPlates` is not missing, the obtained values will be further corrected for plate effects by considering the chosen normalization method. The available options are:

- If `adjustPlates="median"` (median scaling), plates effects are corrected by dividing each measurement by the median value across wells annotated as `sample` in `x$wellAnno`, for each plate and replicate. If the data values are in `log2` scale (`log=TRUE`), the per-plate factor is subtracted from each measurement, instead.
- If `adjustPlates="mean"` (mean scaling), the average in the `sample` wells is consider instead. If the data values are in `log2` scale (`log=TRUE`), the per-plate factor is subtracted from each measurement, instead.
- If `adjustPlates="shorth"` (scaling by the midpoint of the shorth), for each plate and replicate, the midpoint of the `shorth` of the distribution of values in the wells annotated as `sample` is calculated. Then, every measurement is divided by this value (if `log=FALSE`) or subtracted by it (if `log=TRUE`, meaning that data have been log transformed).
- If `adjustPlates="POC"` (percent of control), for each plate and replicate, each measurement is divided by the average of the measurements on the plate positive controls, and multipliplied by 100.
- If `adjustPlates="negatives"`, for each plate and replicate, each measurement is divided by the median of the measurements on the plate negative controls. If the data values are in `log2` scale (`log=TRUE`), the per-plate factor is subtracted from each measurement, instead.
- If `adjustPlates="NPI"` (normalized percent inhibition), each measurement is subtracted from the average of the intensities on the plate positive controls, and this result is divided by the difference between the means of the measurements on the positive and the negative controls.
- If `adjustPlates="Bscore"` (Bscore), for each plate and replicate, the B score method is applied to remove plate effects and row and column biases.

By default, `adjustPlates` is missing.

If `zscore` is not missing, a robust $z$-score for each individual measurement will be determined for each plate and each well by subtracting the overall median and dividing by the overall mad. The overall median and mad are taken by considering the distribution of intensities (over all plates) in the wells whose content is annotated as `sample`. The allowed values for `zscore` ("+" or "-") are used to set the sign of the calculated $z$-scores. For example, with a `zscore="-"` a strong decrease in the signal will be represented by a positive $z$-score, whereas setting `zscore="+"`, such a phenotype will be represented by a negative $z$-score. This option can be set to calculate the results to the commonly used convention.

The arguments `posControls` and/or `negControls` are required for applying the normalization methods based on the control measurements (that is, when `adjustPlates="POC"`, or `adjustPlates="NPI"` or `adjustPlates="negatives"`). `posControls` and `negControls` should be given as a vector of regular expression patterns specifying the name of the positive(s) and negative(s) controls, respectivey, as provided in the plate configuration file (and stored in `x$wellAnno`). The length of these vectors should be equal to the final number of reporters, which in this case is always one. By default, if `posControls` is not given, "pos" will be taken

as the name for the wells containing positive controls. Similarly, if `negControls` is missing, by default "neg" will be considered as the name used to annotate the negative controls. The content of `posControls` and `negControls` will be passed to `regexpr` for pattern matching within the well annotation given in `x$wellAnno` (see examples). The arguments `posControls` and `negControls` are particularly useful in multi-channel data since the controls might be reporter-specific, or after normalizing multi-channel data.

## Value

An object of class `cellHTS`, which is a copy of the argument `x`, plus an additional slot `xnorm` containing the normalized data. This is an array of the same dimensions as `xraw`, except in the dimension corresponding to the number of channels, since the two-channel intensities have been combined into one intensity value.

Moreover, the processing status of the `cellHTS` object is updated in the slot `state` to `x$state["normalized"]=`

Additional outputs may be given if `adjustPlates="Bscore"`. Please refer to the help page of the `Bscore` function.

## Author(s)

Ligia Braz <ligia@ebi.ac.uk>, Wolfgang Huber <huber@ebi.ac.uk>

## See Also

`normalizePlates`, `summarizeChannels`, `Bscore`,

## Examples

```
## Not run:
   datadir <- system.file("DualChannelScreen", package = "cellHTS")
   x <- readPlateData("Platelist.txt", "TwoColorData", path=datadir)
   x <- configure(x, "Plateconf.txt", "Screenlog.txt", "Description.txt", path=datadir)
   table(x$wellAnno)

   ## Define the controls for the different channels:
   negControls=vector("character", length=dim(x$xraw)[4])

   ## channel 1 - gene A
   ## case-insensitive and match the empty string at the beginning and end of a line (tc

   negControls[1]= "(?i)^geneA$"
   ## channel 2 - gene A and geneB
   negControls[2]= "(?i)^geneA$|^geneB$"
   posControls = vector("character", length=dim(x$xraw)[4])
   ## channel 1 - no controls
   ## channel 2 - geneC and geneD
   posControls[2]="(?i)^geneC$|^geneD$"

   writeReport(x, posControls=posControls, negControls=negControls)
   x = normalizeChannels(x, fun=function(x,y) y/x, log=TRUE, adjustPlates="median")
   ## Define the controls for the normalized intensities (only one channel):
   negControls = vector("character", length=dim(x$xnorm)[4])
   ## For the single channel, the negative controls are geneA and geneB
   negControls[1]= "(?i)^geneA$|^geneB$"
   posControls = vector("character", length=dim(x$xnorm)[4])
   ## For the single channel, the negative controls are geneC and geneD
```

```
    posControls[1]="(?i)^geneC$|^geneD$"
    writeReport(x, force=TRUE, plotPlateArgs=list(xrange=c(-3,3)),
         posControls=posControls, negControls=negControls)

## End(Not run)
```

---

`normalizePlates` *Plate-wise data normalization, and data transformation*

---

## Description

Normalization of the data `xraw` in a cellHTS object. This is done separately for each plate, replicate and channel. Optionally, a data transformation such as `log`, and a transformation to $z$-scores can be performed.

## Usage

```
normalizePlates(x, normalizationMethod="median", transform, zscore, posControls,
```

## Arguments

| | |
|---|---|
| `x` | a `cellHTS` object that has already been configured. See details. |
| `normalizationMethod` | a character specifying the normalization method to use for performing the per-plate normalization. Allowed values are `"median"` (default), `"mean"`, `"shorth"`, `"POC"`, `"NPI"`, `"negatives"` and `Bscore`. See details. |
| `transform` | a function that takes a numeric vector and returns a numeric vector of the same length; for example, the logarithm function `log`. |
| `zscore` | indicates if the data should be centered and scaled after normalization and transformation. If missing (default), the data will not be centered and scaled. Otherwise, the value of this argument should be a character string, either "+" or "-", which will be used to set the sign for the calculated $z$-scores. See details. |
| `posControls` | a vector of regular expressions giving the name of the positive control(s). See details. |
| `negControls` | a vector of regular expressions giving the name of the negative control(s). See details. |
| `...` | Further arguments that get passed on to the function implementing the normalization method chosen by `normalizationMethod`. Currently, this is only used for `Bscore`. |

## Details

The normalization is performed in a plate-by-plate fashion.

- If `normalizationMethod="median"` (median scaling), plates effects are corrected by dividing each measurement by the median value across wells annotated as `sample` in `x$wellAnno`, for each plate and replicate.
- If `normalizationMethod="mean"` (mean scaling), the average in the `sample` wells is consider instead.

- If `normalizationMethod="shorth"` (scaling by the midpoint of the shorth), for each plate and replicate, the midpoint of the `shorth` of the distribution of values in the wells annotated as `sample` is calculated. Then, every measurement is divided by this value.
- If `normalizationMethod="POC"` (percent of control), for each plate and replicate, each measurement is divided by the average of the measurements on the plate positive controls, and multiplied by 100.
- If `normalizationMethod="negatives"` (scaling by the negative controls), for each plate and replicate, each measurement is divided by the median of the measurements on the plate negative controls.
- If `normalizationMethod="NPI"` (normalized percent inhibition), each measurement is subtracted from the average of the intensities on the plate positive controls, and this result is divided by the difference between the means of the measurements on the positive and the negative controls.
- If `normalizationMethod="Bscore"` (B score), for each plate and replicate, the B score method is applied to remove plate effects and row and column biases. NOTE: if the argument 'transform' is given, the B score method is applied AFTER data transformation.

If `transform` is not missing, the chosen data transformation is applied. Most commonly, this option can be used to apply a log transformation.

If `zscore` is not missing, a robust $z$-score for each individual measurement will be determined for each plate and each well by subtracting the overall `median` and dividing by the overall `mad`. These are taken by considering the distribution of intensities (over all plates) in the wells whose content is annotated as `sample`. The allowed values for `zscore` ("+" or "-") are used to set the sign of the calculated $z$-scores. For example, with a `zscore="-"` a strong decrease in the signal will be represented by a positive $z$-score, whereas setting `zscore="+"`, such a phenotype will be represented by a negative $z$-score. This option can be set to calculate the results to the commonly used convention.

The arguments `posControls` and `negControls` are required for applying the normalization methods based on the control measurements (that is, when `normalizationMethod="POC"`, or `normalizationMethod="NPI"`, or `normalizationMethod="negatives"`). `posControls` and `negControls` should be given as a vector of regular expression patterns specifying the name of the positive(s) and negative(s) controls, respectivey, as provided in the plate configuration file (and stored in `x$wellAnno`). The length of these vectors should be equal to the number of reporters used in the screen (`dim(x$xraw)[4]`) or to `dim(x$xnorm)[4]`, in case `x` contains multi-channel data that have been normalized by combining the values from two or more channels. By default, if `posControls` is not given, *pos* will be taken as the name for the wells containing positive controls. Similarly, if `negControls` is missing, by default *neg* will be considered as the name used to annotate the negative controls. The content of `posControls` and `negControls` will be passed to `regexpr` for pattern matching within the well annotation given in `x$wellAnno` (see examples for `summarizeChannels`). The arguments `posControls` and `negControls` are particularly useful in multi-channel data since the controls might be reporter-specific, or after normalizing multi-channel data.

## Value

An object of class `cellHTS`, which is a copy of the argument `x`, plus an additional slot `xnorm` containing the normalized data. This is an array of the same dimensions as `xraw`.

Moreover, the processing status of the `cellHTS` object is updated in the slot `state` to `x$state["normalized"]=`

Additional outputs may be given if `adjustPlates="Bscore"`. Please refer to the help page of the `Bscore` function.

### Author(s)

Ligia Braz <ligia@ebi.ac.uk>, Wolfgang Huber <huber@ebi.ac.uk>

### See Also

Bscore, summarizeChannels

### Examples

```
data(KcViabSmall)
x1 = normalizePlates(KcViabSmall, normalizationMethod="median", zscore="-")
## Not run:
x2 = normalizePlates(KcViabSmall, normalizationMethod="Bscore", zscore="-")

## End(Not run)
```

---

## oneRowPerId *Rearrange dataframe entries such that there is exactly one row per ID.*

---

### Description

Rearrange dataframe entries such that there is exactly one row per ID. The IDs are taken from the argument `ids` and are matched against the first column of `x`. If an ID is missing in `x[,1]`, a row with `NA` values is inserted. If an ID occurs multiple times in `x[,1]`, rows are collapsed into characters of comma-separated values.

### Usage

```
oneRowPerId(x, ids)
```

### Arguments

| | |
|---|---|
| `x` | dataframe. |
| `ids` | character vector. |

### Value

A dataframe whose rows correspond 1:1 to `ids`.

### Author(s)

W. Huber <huber@ebi.ac.uk>, Ligia Pedroso Braz <ligia@ebi.ac.uk>

### Examples

```
x = data.frame(ids=I(c("a", "a", "c")), val=11:13)
oneRowPerId(x, letters[1:3])
```

`plotPlateLibrary` *Plate plot of the raw data of the four consecutive 96-well plates of a*

## Description

Given a `cellHTS` object with data from an assay where every set of four consecutive 96-well plates was combined into a 384-well plate, this function plots the raw intensities of a chosen 384-well assay plate distributed according to the 96-well plate format.

## Usage

```
plotPlateLibrary(x, whichPlate = 1, whichChannel = 1, plotSd = TRUE, plotPlateAr
```

## Arguments

| | |
|---|---|
| `x` | a cellHTS object containing the slot `libPlate` with the 96-well plate identifiers (see `getLibraryPlate`). |
| `whichPlate` | a number indicating the 384-well plate that we want to examine. (By default, it considers the first plate, `whichPlate = 1`). |
| `whichChannel` | a number indicating the channel that we want to consider. (By default, the first channel is considered: `whichChannel = 1`) |
| `plotSd` | a logical value indicating whether the standard deviation across replicates should be plotted (default is `plotSd = TRUE`. |
| `plotPlateArgs` | optional argument. If given, should be a list with parameters for the plate plots. See details. |

## Details

The cellHTS object `x` contains data from a screening experiment where every set of four consecutive 96-well plates was combined into a 384-well plate. The plate identifiers for the 96-well plates are given in the slot `libPlate`, obtained by `getLibraryPlate`.

Given the channel specified by `whichChannel` and the 384-well plate number `whichPlate`, the function plots the raw intensities for each replicate in both the 96-well and the 384-well plate format. If `plotSd = TRUE`, the standard deviation across replicates is also plotted.

The following elements are recognized for `plotPlateArgs` and passed on to `plotPlate`: `sdcol`, the color scheme for the standard deviation plate plot, `sdrange`, the sd range to which the colors are mapped, `xcol`, the color scheme for the intensity plate plot, `xrange`, the intensity range to which the colors are mapped. If an element is not specified, default values are used.

## Author(s)

Ligia Braz `<ligia@ebi.ac.uk>`

## See Also

`plotPlate`, `getLibraryPlate`

### Examples

```
data(KcViabSmall)
x <- getLibraryPlate(KcViabSmall)
plotPlateLibrary(x, whichPlate=2, plotSd=TRUE)
```

---

## plotSpatialEffects *Plate plot with the row and column offsets estimated by the B score*

---

### Description

The function plots the per-plate row and column effects estimated by the B score method.

### Usage

```
plotSpatialEffects(x, whichChannel = 1, plateRange)
```

### Arguments

| | |
|---|---|
| x | a cellHTS object that has already been normalized using the B score method (see details). |
| whichChannel | a numeric value giving the channel of x to plot. |
| plateRange | a numeric vector giving the plate numbers to plot. If missing, the function considers all the plates. |

### Details

The function plots the plate plots displaying the row and column offsets (stored in slot rowcol.effects of the cellHTS object x) within the plates in plateRange, and for channel whichChannel, as determined by the B score method. Before plotting the spatial offsets, the values within the chosen channel (whichChannel) are transformed in order to be confined in the range $[0, 1]$, as follows:

$$y^t = \frac{(y - min(y))}{max(y) - min(y)}$$

Here, $y^t$ are the transformed values, and $y$ the estimated spatial effects. The maximum and the minimum values are calculated using all of the values in x$rowcol.effects[,,whichChannel].

### Author(s)

Ligia P. Bras <ligia@ebi.ac.uk>

### See Also

plotPlate, Bscore, normalizePlates, summarizeChannels

### Examples

```
data(KcViabSmall)
x = normalizePlates(KcViabSmall, normalizationMethod="Bscore",
                    save.model = TRUE)
## see plate plots with the row and column estimated offsets for
## plates 1 and 3:
plotSpatialEffects(x, plateRange=c(1,3))
```

---

## print.cellHTS *Printing cellHTS objects*

---

### Description

Print an object of the class 'cellHTS'.

### Usage

```
## S3 method for class 'cellHTS'
print(x, ...)
```

### Arguments

| | |
|---|---|
| x | object of class cellHTS. |
| ... | optional arguments to print methods. |

### Details

Shows the information about the cellHTS object x, namely, its name, state, and the number of plates, wells, replicates and channels.

### Author(s)

Wolfgang Huber <huber@ebi.ac.uk>, Ligia Braz <ligia@ebi.ac.uk>

### Examples

```
data(KcViabSmall)
print(KcViabSmall)
```

readPlateData *Read a collection of plate reader data files*

## Description

Reads a collection of plate reader data files into a data.frame. The names of the files, plus additional information (plate number, repeat number) is expected in a tab-delimited table specified by the argument x.

## Usage

```
readPlateData(filename, path=dirname(filename), name, importFun, verbose=TRUE, p
```

## Arguments

| | |
|---|---|
| filename | the name of the file table (see details). This argument is just passed on to the read.table function, so any of the valid argument types for read.table are valid here, too. |
| name | a character of length 1 with the experiment name. |
| path | a character of length 1 indicating the path in which to find the plate reader files. By default, it can extract the path from filename. |
| importFun | a function that should be used to read each plate result file. The default function works for plate reader data files. See details. |
| verbose | a logical value, if TRUE, the function reports some of its intermediate progress. |
| plateType | (deprecated argument) a character of length 1 giving the format of the plate: "96" for 96-well plate format, or "384" for a 384-well plate format. |

## Details

The file table is expected to be a tab-delimited file with at least three columns, and column names Filename, Plate, and Replicate. The contents of the columns Plate and Replicate are expected to be integers. Further columns are allowed.

We distinguish between *plates* and *plate result file*. A plate result file contains the measurements results for all replicates and all channels of a plate, which is the physical carrier of the reagents.

importFun can be used to define other functions to import other data files, such as flow cytometry data files, etc. The importFun function should receive as an input the name of a result plate file to read, and return a list with two components:

- The first component should be a 'data.frame' with the following slots:
  - well, a character vector with the well identifier in the plate.
  - val, the intensity values measured at each well.
- The second component of this list should be a character vector containing a copy of the imported input data file (such as the output of readLines). It should be suitable to be used as input for writeLines, since it will be used to reproduce the intensity files that are linked in the HTML quality reports generated by writeReport.

For example, to import plate data files from EnVision plate reader, set importFun=getEnVisionRawData or importFun=getEnvisionCrosstalkCorrectedData. See function getEnVisionRawData.

## Value

An object of class "cellHTS", which is currently implemented as a list with elements

| | |
|---|---|
| name | copy of the input argument name |
| xraw | an array of dimension plateSize x number of plates x number of replicates x number of channels, containing the imported measurement data. |
| pdim | a numeric vector of length 2 containing the number of rows and columns in a plate. The product of these two numbers is the first dimension of xraw. This corresponds to the plate format used in the screen, and it is automatically determined from the plate result files. The allowed formats are 96-well or 384-well plates. |
| batch | an integer vector with the batch number (1, 2, ...) for each plate. Its length corresponds to the second dimension of of xraw. |
| plateList | a data.frame containing what was read from input file x, plus a column status of type character: it contains the string "OK" if the data import appeared to have gone well, and the respective error or warning message otherwise. |
| intensityFiles | a list, where each component contains a copy of the imported input data files. Its length corresponds to the number of rows of plateList. |
| state | a logical vector representing the processing status of the object. |

## Author(s)

W. Huber <huber@ebi.ac.uk>, Ligia Braz <ligia@ebi.ac.uk>

## See Also

getEnVisionRawData

## Examples

```
datadir <- system.file("KcViabSmall", package = "cellHTS")
x <- readPlateData("Platelist.txt", "KcViabSmall", path=datadir)

## To read data files obtained from an EnVision plate reader:
datadir <- system.file("EnVisionExample", package = "cellHTS")
x <- readPlateData("platelist.txt", "EnVisionEx",
      importFun=getEnVisionRawData, path=datadir)

   ## to get the cross talk corrected data:
   y <- readPlateData("platelist.txt", "EnVisionEx",
       importFun=getEnVisionCrosstalkCorrectedData, path=datadir)
```

`screenMatch` *Matching the gene annotation of two screens*

## Description

Match the annotation of two `cellHTS` objects in order to find the common gene-perturbing reagents

## Usage

```
screenMatch(screens, ids)
```

## Arguments

| | |
|---|---|
| `screens` | a list of annotated cellHTS objects. |
| `ids` | a character vector of length two giving, for each cellHTS object, the name of the column of slot 'geneAnno' that should be used for the annotation IDs. See details. |

## Details

By default, if `ids` is missing, the column `GeneID` of the slot `geneAnno` of each of the `cellHTS` objects in `screens` is taken for the annotation IDs when comparing the two data sets.

## Value

A list with two components:

| | |
|---|---|
| `p.overlap` | a vector giving the proportion of overlap in the screens' annotation. |
| `isInBoth` | a list of logical vectors, each of which with length equal to the product between `nr. Well` and `nr. Plates` in each screen, indicating whether the respective gene-perturbing reagent of that screen is also present in the other. |

## Author(s)

Ligia Braz <ligia@ebi.ac.uk>

## Examples

```
## Just for exemplification purposes, we consider the complete genome-wide screen "Kc
## and its first 3 plates ("KcViabSmall"):
data(KcViab)
data(KcViabSmall)
screens <- list(KcViab, KcViabSmall)
out <- screenMatch(screens)
out$p.overlap
sapply(out$isInBoth, sum)
sapply(1:2, function(z) table(screens[[z]]$geneAnno$Plate[out$isInBoth[[z]]]))
```

summarizeChannels *Normalization and transformation of dual-channel data*

## Description

Normalizes and/or transforms dual-channel data xraw of a cellHTS object by applying the function defined in fun.

## Usage

```
summarizeChannels(x,
    fun = function(r1, r2, thresh) ifelse(r1>thresh, log2(r2/r1), as.numeric(NA)
    adjustPlates, zscore, ...)
```

## Arguments

| | |
|---|---|
| x | a cellHTS object that has been configured. |
| fun | a user-defined function for the two channel summarization. fun takes two numeric vectors and returns a numeric vector of the same length. The default is to take the log2-ratio between the second and first channels, with a threshold on r1 shown above in the *Usage* section that should be set by the user. |
| adjustPlates | scalar character string indicating the normalization method to apply to adjust for plate-to-plate variations (and possibly well-to-well variations). This is done *before* applying fun. Allowed values are "median", "mean", "shorth", "POC", "NPI", "negatives" and Bscore. If adjustPlates is missing (the default), no plate-wise correction will be performed. |
| zscore | indicates if the $z$-scores should be determined after normalization and transformation. If missing (default), the data will not be scored. Otherwise, it should be a character string, either "+" or "-", specifying the sign to use for the $z$-scores. |
| ... | Further arguments that get passed on to the function implementing the normalization method chosen by adjustPlates. See the *Details* section and the normalizePlates function. |

## Details

For each plate and replicate of a two-color experiment, the function defined in fun is applied to relate the intensity values in the two channels of the cellHTS object. The default is to take the log2-ratio between the second and first channels, with a threshold on r1 (see the *Usage* section). This threshold should be ajusted by the user according to the data. For an example, see the *Examples* section.

If adjustPlates is not missing, the values for each channel will be corrected for plate effects *before* applying fun, by considering the chosen normalization method. The available options are adjustPlates="median" (median scaling), adjustPlates="mean" (mean scaling), adjustPlates="shorth" (scaling by the midpoint of the shorth), adjustPlates="POC" (percent of control), adjustPlates="negatives" (scaling by the average on the negative controls), adjustPlates="NPI" (normalized percent inhibition) and adjustPlates="Bscore" (B score method). For more details about these normalization options, please refer to normalizePlates. By default, adjustPlates is missing.

If zscore is not missing, a robust $z$-score is calculated based on the channel-summarized measurements. The $z$-score for each individual measurement will be determined for each plate and each

well by subtracting the overall median and dividing by the overall mad. The allowed values for zscore ("+" or "-") are used to set the sign of the calculated $z$-scores. See summarizeReplicates for more details.

## Value

An object of class cellHTS, which is a copy of the argument x, plus an additional slot xnorm containing the normalized data. This is an array of the same dimensions as xraw, except in the dimension corresponding to the number of channels, since the two-channel intensities have been combined into one intensity value.

Moreover, the processing status of the cellHTS object is updated in the slot state to x$state["normalized"]=

Additional outputs may be given if adjustPlates="Bscore". Please refer to the help page of the Bscore function.

## Author(s)

Ligia Braz <ligia@ebi.ac.uk>, Wolfgang Huber <huber@ebi.ac.uk>

## See Also

normalizePlates, Bscore, summarizeReplicates

## Examples

```
## Not run:
   datadir <- system.file("DualChannelScreen", package = "cellHTS")
   x <- readPlateData("Platelist.txt", "TwoColorData", path=datadir)
   x <- configure(x, "Plateconf.txt", "Screenlog.txt", "Description.txt", path=datadir)
   table(x$wellAnno)

   ## Define the controls for the different channels:
   negControls=vector("character", length=dim(x$xraw)[4])

   ## channel 1 - gene A
   ## case-insensitive and match the empty string at the beginning and end of a line (tc

   negControls[1]= "(?i)^geneA$"
   ## channel 2 - gene A and geneB
   negControls[2]= "(?i)^geneA$|^geneB$"
   posControls = vector("character", length=dim(x$xraw)[4])
   ## channel 1 - no controls
   ## channel 2 - geneC and geneD
   posControls[2]="(?i)^geneC$|^geneD$"

   writeReport(x, posControls=posControls, negControls=negControls)
   ## In this example, we first normalize each channel separately by plate median scalin
   ## Then, we define a low intensity threshold for the measurements in the constitutive
   ## which will be set to the 5
   x = summarizeChannels(x, fun = function(r1, r2,
            thresh=quantile(r1, probs=0.05, na.rm=TRUE)) ifelse(r1>thresh, log2(r2/r1),
     adjustPlates="median")
   ## Note that the plate median scaling is applied to each channel, prior to channel su
   ## Define the controls for the normalized intensities (only one channel):
   negControls = vector("character", length=dim(x$xnorm)[4])
   ## For the single channel, the negative controls are geneA and geneB
```

```
    negControls[1]= "(?i)^geneA$|^geneB$"
    posControls = vector("character", length=dim(x$xnorm)[4])
    ## For the single channel, the negative controls are geneC and geneD
    posControls[1]="(?i)^geneC$|^geneD$"
    writeReport(x, force=TRUE, plotPlateArgs=list(xrange=c(-3,3)),
         posControls=posControls, negControls=negControls)

## End(Not run)
```

---

`summarizeReplicates` *Summarizes between normalized replicate values given in a cellHTS*

---

## Description

Summarizes the normalized (and possibly already scored) replicate values given in a `cellHTS` object, and calculates a single $z$-score value for each probe.

Currently this function is implemented only for single-color data.

## Usage

```
summarizeReplicates(x, zscore, summary = "min")
```

## Arguments

| | |
|---|---|
| `x` | a cellHTS object that has already been normalized (see details). |
| `zscore` | indicates if the replicate values should be centered and scaled. If missing (default), the data will not be centered and scaled. Otherwise, the value of this argument should be a character string, either "+" or "-", specifying the sign to use for the calculated $z$-scores (see details). |
| `summary` | a character string indicating how to summarize between replicated measurements. One of "min" (default), "mean", "max", "rms", "closestToZero", or "FurthestFromZero" can be used (see details). |

## Details

Given the normalized values given in the slot `xnorm` of `x`, a single $z$-score is calculated for each probe.

The argument `zscore` indicates the state of the normalized replicate measurements: if `zscore` is missing, it is assumed that the replicates have been scored, by calling `normalizePlates` with the argument `zscore` equal to "-" or "+"; Otherwise, `zscore` should be given, so that a robust $z$-score is calculated for each plate and each well by subtracting the overall median and dividing by the overall mad. The overall median and mad are taken by considering the distribution of intensities (over all plates) in the wells whose content is annotated as `sample`. The allowed values for `zscore` ("+" or "-") are used to set the sign of the calculated $z$-scores. For example, with a `zscore="-"` a strong decrease in the signal will be represented by a positive $z$-score, whereas setting `zscore="+"`, such a phenotype will be represented by a negative $z$-score. This option can be set to calculate the results to the commonly used convention.

Finally, a single $z$-score per probe is calculated by summarizing between scored replicates. If `summary="mean"`, the average of replicate values is considered; if `summary="max"`, then

the maximum of replicate intensities is taken; if `summary="min"`, the minimum is considered, instead (conservative); if `summary="rms"`, the square root of the mean squared value of the replicates (root mean square) is taken as a summary function; if `summary="closestToZero"`, the value closest to zero is taken as a summary (ueful when both sides of the distribution of z-score values are of interest); if `summary="furthestFromZero"`, the value furthest from zero is taken as a summary (ueful when both sides of the distribution of z-score values are of interest)

### Value

An object of class `cellHTS`, which is a copy of the argument `x` plus the slot `score`, a numeric vector containing the z-factor for each well in every plate. The length of this vector is therefore equal to the product between the plateSize and the number of plates. Moreover, the processing status of the `cellHTS` object is updated in the slot `state` to `state["scored"]= TRUE`.

### Author(s)

W. Huber <huber@ebi.ac.uk>, Ligia Braz <ligia@ebi.ac.uk>

### See Also

`normalizePlates`, `summarizeChannels`

### Examples

```
data(KcViabSmall)
x <- normalizePlates(KcViabSmall, normalizationMethod="median")
x <- summarizeReplicates(x, zscore="-", summary="min")
```

---

## write.tabdel *Wrapper to function 'write.table' used to write data to a tab-delimited*

---

### Description

Wrapper for the function `write.table` to write data to a tab-delimited file.

### Usage

```
write.tabdel(...)
```

### Arguments

| | |
|---|---|
| `...` | arguments that get passed on to the function `write.table`. |

### Details

A trivial function, which we have included for convenience.

### Value

The name of the file that was written.

### Author(s)

Ligia Braz <ligia@ebi.ac.uk>

### See Also

write.table

### Examples

```
data(KcViabSmall)
x <- KcViabSmall
## determine the ratio between each well and the plate median
y <- array(as.numeric(NA), dim=dim(x$xraw))
nrWell <- dim(x$xraw)[1]
for(p in 1:(dim(x$xraw)[2])) {
  samples <- (x$wellAnno[(1:nrWell)+nrWell*(p-1)]=="sample")
  y[, p, , ] <- apply(x$xraw[, p, , , drop=FALSE], 3:4,
                      function(w) w/median(w[samples], na.rm=TRUE))
}
y <- signif(y, 4)
out <- matrix(y, nrow=prod(dim(y)[1:2]), ncol=dim(y)[3:4])
out <- cbind(x$geneAnno, x$wellAnno, out)
colnames(out) <- c(names(x$geneAnno), "wellAnno",
   sprintf("Well/Median_r%d_ch%d",
           rep(1:dim(y)[3], dim(y)[4]), rep(1:dim(y)[4], each=dim(y)[3])))
write.tabdel(out, file = tempfile())
```

---

## writeReport *Create a directory with HTML pages of linked tables and plots*

---

### Description

Creates a directory with HTML pages of linked tables and plots documenting the contents of a cellHTS object.

### Usage

```
writeReport(x,
  outdir=file.path(getwd(), x$name),
  force=FALSE,
  plotPlateArgs=FALSE,
  imageScreenArgs=NULL,
  progressReport = interactive(),
  posControls,
  negControls)
```

### Arguments

| | |
|---|---|
| x | a cellHTS object. |
| outdir | a character of length 1 with the name of a directory where to write the report HTML file and images. If the directory does not exist, it is created. If it exists and is not empty, then the behaviour depends on the value of force. |

force — a logical value, determines the behaviour of the function if outdir exists and is not empty. If force is TRUE, the function overwrites (removes and recreates) outdir, otherwise it casts an error.

plotPlateArgs — either a list with parameters for the plate plots of the per plate quality report pages, or a logical scalar with values FALSE or TRUE. If FALSE, the plate plots are omitted, this option is here because the production of the plate plots takes a long time. See details.

imageScreenArgs — a list with parameters for the function imageScreen. See details.

progressReport — a logical, should a progress report window be displayed?

posControls — a list or vector of regular expressions specifying the name of the positive controls. See details.

negControls — a vector of regular expressions specifying the name of the negative controls. See details.

## Details

The following elements are recognized for plotPlateArgs and passed on to plotPlate: sdcol, the color scheme for the standard deviation plate plot, sdrange, the sd range to which the colors are mapped, xcol, the color scheme for the intensity plate plot, xrange, the intensity range to which the colors are mapped. If an element is not specified, default values are used.

The following elements are recognized for imageScreenArgs and passed on to imageScreen: ar, aspect ratio, zrange, range, map, logical value indicating whether tooltips with the annotation should be added to the plot (default value is FALSE), anno, gene annotation for the image map.

posControls and negControls should be given as a vector of regular expression patterns specifying the name of the positive(s) and negative(s) controls, respectivey, as provided in the plate configuration file (and stored in x$wellAnno). The length of these vectors should be equal to the number of reporters used in the screen (dim(x$xraw)[4] or to dim(x$xnorm)[4], in case x contains multi-channel data that have been normalized by combining the values from two or more channels). By default, if posControls is not given, "pos" will be taken as the name for the wells containing positive controls. Similarly, if negControls is missing, by default "neg" will be considered as the name used to annotate the negative controls. The content of posControls and negControls will be passed to regexpr for pattern matching within the well annotation given in x$wellAnno (see examples). If no controls are available for a given channel, use "" or NA for that channel. For example, posControls = c("", "(?i)^diap$") means that channel 1 has no positive controls, while "diap" is the positive control for channel 2.

The arguments posControls and negControls are particularly useful in multi-channel data since the controls might be reporter-specific, or after normalizing multi-channel data.

In case of a two-way assay, where two types of "positive" controls are used in the screen ("activators" and "inhibitors"), posControls should be defined as a list with two components (called act and inh), each of which should be vectors of regular expressions of the same length as the current number of reporters (as explained above).

By default, tooltips doing the mapping between the probe annotation and the plate wells are not added to the plate plots and to the overall screen plot. If the cellHTS object x is annotated, the probe annotation is based on the information contained whether in x$geneAnno$GeneSymbol, or x$geneAnno$GeneID, if the former is missing. Otherwise, the mapping simply uses the well identifiers.

### Value

The function is called for its side-effect. It returns a character with the full path and name of the report index file, this is an HTML file which can be read by a web browser.

### Author(s)

Ligia Braz <ligia@ebi.ac.uk>, Wolfgang Huber <huber@ebi.ac.uk>

### See Also

plotPlate, imageScreen

### Examples

```
data(KcViabSmall)
## pCtrls <- c("pos")
## nCtrls <- c("neg")
## or for safety reasons (not a problem for the current well
## annotation, however)
## pCtrls <- c("^pos$")
## nCtrls <- c("^neg$")
## writeReport(KcViabSmall, posControls=pCtrls, negControls=nCtrls)
## same as
## writeReport(KcViabSmall)
## Not run:
x <- normalizePlates(KcViabSmall, normalizationMethod="median",zscore="-")
x <- summarizeReplicates(x, summary="min")
writeReport(x, force=TRUE, plotPlateArgs = list(), imageScreenArgs=list(zrange=c(-4,4

## End(Not run)
    ## to turn on the tooltips in the overall screen image plot:
    ## writeReport(x, force=TRUE, plotPlateArgs = list(), imageScreenArgs = list(zrange=c
```

---

## writeTab — *Write the data from a cellHTS object to a tab-delimited file*

---

### Description

Write the data from a cellHTS object to a tab-delimited file.

### Usage

```
writeTab(x, ...)
## S3 method for class 'cellHTS'
writeTab(x, file=paste(x$name, "txt", sep="."), ...)
```

### Arguments

| | |
|---|---|
| x | a cellHTS object. |
| file | the name of the output file. |
| ... | ignored. |

## Details

This function is a wrapper for function `write.table` to write the data from a `cellHTS` object to a tab-delimited file.

## Value

The name of the file that was written.

## Author(s)

Wolfgang Huber <huber@ebi.ac.uk>, Ligia Braz <ligia@ebi.ac.uk>

## Examples

```
data(KcViabSmall)
writeTab(KcViabSmall, file=tempfile())
```

# Index