Package 'MAPFX'

December 6, 2024

Title MAssively Parallel Flow cytometry Xplorer (MAPFX): A Toolbox for Analysing Data from the Massively-Parallel Cytometry Experiments

Version 1.2.0

Description MAPFX is an end-to-end toolbox that pre-processes the raw data from MPC experiments (e.g., BioLegend's LEGENDScreen and BD Lyoplates assays), and further imputes the 'missing' infinity markers in the wells without those measurements. The pipeline starts by performing background correction on raw intensities to remove the noise from electronic baseline restoration and fluorescence compensation by adapting a normal-exponential convolution model. Unwanted technical variation, from sources such as well effects, is then removed using a log-normal model with plate, column, and row factors, after which infinity markers are imputed using the informative backbone markers as predictors. The completed dataset can then be used for clustering and other statistical analyses. Additionally, MAPFX can be used to normalise data from FFC assays as well.

Depends R (>= 4.4.0)

License GPL-2

Encoding UTF-8

RoxygenNote 7.3.1

Roxygen list(markdown = TRUE)

biocViews Software, FlowCytometry, CellBasedAssays, SingleCell, Proteomics, Clustering

Imports flowCore, Biobase, stringr, uwot, iCellR, igraph, ggplot2, RColorBrewer, Rfast, ComplexHeatmap, circlize, glmnetUtils, e1071, xgboost, parallel, pbapply, reshape2, gtools, utils, stats, cowplot, methods, grDevices, graphics

Suggests BiocStyle, knitr, rmarkdown, testthat

VignetteBuilder knitr

NeedsCompilation no

URL https://github.com/HsiaoChiLiao/MAPFX

BugReports https://github.com/HsiaoChiLiao/MAPFX/issues

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

git_branch RELEASE_3_20

git_last_commit 3a93e18

git_last_commit_date 2024-10-29

Repository Bioconductor 3.20

Date/Publication 2024-12-05

Author Hsiao-Chi Liao [aut, cre] (<https://orcid.org/0000-0002-9586-1246>),
Agus Salim [ctb],
infinityFlow [ctb]

Maintainer Hsiao-Chi Liao <chelsea.acad@gmail.com>

Contents

---

bkc_bkb *Background correction for the backbone markers*

Description

This function has been designed to do background correction for the backbone markers by using normal-exponential convolution model.

Usage

```
bkc_bkb(
  paths,
  bkb.v,
  MPC,
  bkb.upper.quantile = 0.9,
  bkb.lower.quantile = 0.1,
  bkb.min.quantile = 0.01,
  plots = TRUE
)
```

Arguments

paths	a vector of characters of paths to store intput, intermediary results, outputs...
bkb.v	a vector of the names of the backbone markers (MUST match to the names in the FCS file).
MPC	if the data is from MPC experiments, set MPC = TRUE. Setting FALSE represents data from the fluorescence flow cytometry (FFC) assay.
bkb.upper.quantile	the cut-off (default = 0.9) for selecting cells used for estimating the parameter of signal.
bkb.lower.quantile	the cut-off (default = 0.1) for selecting cells used for estimating the parameters of noise.
bkb.min.quantile	the cut-off (default = 0.01) for omitting the cells with the smallest values to minimise the impact of outliers.
plots	logical; if TRUE (default), produce scatter plots for pre- and post- background adjusted backbone markers (calibrated values on y-axis and raw values on x-axis).

Details

Generating the calibrated measurements and save to medpara_bkc.bkb_no.bkcPhy_mt.rds file, and visualising the result with the scatter plots in the output directory.

Value

Background noise corrected backbone markers and graphs if specified

Author(s)

Hsiao-Chi Liao and Agus Salim

---

bkc_pe *Background correction for the well-specific markers (PE)*

Description

This function has been designed to do background correction for the well-specific markers (PE) by using normal-exponential convolution model.

Usage

```
bkc_pe(paths, pe.lower.quantile = 0.1, pe.min.quantile = 0.01, plots = TRUE)
```

Arguments

paths	a vector of characters of paths to store intput, intermediary results, outputs...
pe.lower.quantile	the cut-off (default = 0.1) for selecting cells used for estimating the parameters of noise for infinity markers.
pe.min.quantile	the cut-off (default = 0.01) for omitting the cells with the smallest values to minimise the impact of outliers for infinity markers.
plots	logical; if TRUE (default), produce scatter plots for pre- and post- background adjusted infinity markers (calibrated values on y-axis and raw values on x-axis).

Details

Generating the calibrated measurements and save to bkc.pe_mt.rds file, and visualising the result with the scatter plots in the output directory.

Value

Background noise corrected infinity markers and graphs if specified

Author(s)

Hsiao-Chi Liao and Agus Salim

---

cluster.analysis *Cluster analysis with normalised backbone measurements and the complete dataset*

Description

This function has been designed to perform cluster analysis for the normalised backbone measurements and the complete dataset which includes the normalised backbone measurements and the imputed well-specific markers.

Usage

```
cluster.analysis(paths, bkb.v, yvar = "Legend", control.wells, plots = TRUE)
```

Arguments

paths	a vector of characters of paths to store intput, intermediary results, outputs...
bkb.v	a vector of the names of the backbone markers (MUST match to the names in the FCS file).
yvar	the name of the well-specific marker in the FCS files (e.g., "Legend").
control.wells	the well label of the control wells, including the autofluorescence and the isotype controls (format: plate_well, e.g., P1_A01)
plots	logical; if TRUE (default), produce an UMAP embedding plot from the normalised backbone markers and the imputed infinity markers to visualise the structure of the biological clusters.

Details

Updating the metadata for cells in the fcs_metadata_df.rds file, adding the information of the biological clusters from the clean and complete dataset, and visualising the result with the scatter plots in the output directory.

Value

Metadata for cells with group labels from the cluster analysis

Author(s)

Hsiao-Chi Liao

`cluster.analysis.bkbOnly` *Cluster analysis with normalised backbone measurements*

Description

This function has been designed to perform cluster analysis for the normalised backbone measurements.

Usage

```
cluster.analysis.bkbOnly(paths, bkb.v, plots = TRUE)
```

Arguments

`paths`	a vector of characters of paths to store intput, intermediary results, outputs...
`bkb.v`	a vector of the names of the backbone markers (MUST match to the names in the FCS file).
`plots`	logical; if TRUE (default), produce an UMAP embedding plot from the normalised backbone markers to visualise the structure of the biological clusters.

Details

Updating the metadata for cells in the fcs_metadata_df.rds file, adding the information of the biological clusters from the clean and complete dataset, and visualising the result with the scatter plots in the output directory.

Value

Metadata for cells with group labels from the cluster analysis

Author(s)

Hsiao-Chi Liao

fcs_to_rds *Converting FCS files to RDS files*

Description

This function has been designed to convert the raw FCS files to data matrix and export to RDS files.

Usage

```
fcs_to_rds(paths, file_meta, yvar)
```

Arguments

paths	a vector of characters of paths to store intput, intermediary results, outputs...
file_meta	if the file names of the FCS files are in the specified format, set file_meta="auto"; otherwise set file_meta="usr" and provide "filename_meta.csv" in FCSpath.
yvar	the name of the well-specific marker in the FCS files (e.g., "Legend").

Details

Generating fcs_metadata_df.rds and fcs_rawInten_mt.rds files in the output directory.

Value

Raw protein intensities and the corresponding metadata from MPC experiments

Author(s)

Hsiao-Chi Liao

fcs_to_rds_bkb *Converting FCS files to RDS files (for the case without exploratory markers)*

Description

This function has been designed to convert the raw FCS files to data matrix and export to RDS files.

Usage

```
fcs_to_rds_bkb(paths, file_meta, MPC)
```

Arguments

paths	a vector of characters of paths to store intput, intermediary results, outputs...
file_meta	if the file names of the FCS files are in the specified format, set file_meta="auto"; otherwise set file_meta="usr" and provide "filename_meta.csv" in FCSpath.
MPC	if the data is from MPC experiments, set MPC = TRUE. Setting FALSE represents data from the fluorescence flow cytometry (FFC) assay.

Details

Generating fcs_metadata_df.rds and fcs_rawInten_mt.rds files in the output directory.

Value

Raw protein intensities and the corresponding metadata from FFC experiments

Author(s)

Hsiao-Chi Liao

`imputation_bkb.predictors` *Imputing the unmeasured well-specific markers with regression models*

Description

This function has been designed to impute/predict the unmeasured well-specific markers with regression models.

Usage

```
imputation_bkb.predictors(
  paths,
  chans,
  yvar = "Legend",
  cores = 4L,
  models.use,
  extra_args_regression_params,
  prediction_events_downsampling = NULL,
  impu.training,
  plots = TRUE
)
```

Arguments

`paths`	a vector of characters of paths to store intput, intermediary results, outputs...
`chans`	a vector of the names of the backbone markers (MUST match to the names in the FCS file).
`yvar`	the name of the well-specific marker in the FCS files (has been changed to "Legend" in the first function).
`cores`	the number of cores used to perform parallel computation (default = 8L).
`models.use`	a vector of the names of the models used for imputation (an example: `c("LM", "LASSO3", "SVM", "XGBoost")`). The length of the vector is arbitrary.
`extra_args_regression_params`	a list of the lists of the parameters for running regression models.

`prediction_events_downsampling`
: default = NULL (not doing subsampling). How many cells per well you want to have the imputation? (must be less than or equal to a half as we won't get the prediction for cells in the training set).

`impu.training`
: logical; if FALSE (default), not impute the training set (the dataset used to train the imputation models).

`plots`
: logical; if TRUE (default), visualise the distribution of R-sq from each infinity marker.

Details

This function returns the object of imputation of the unmeasured well-specific markers. In the output directory, the imputations are saved to predictions.Rds file. Visualisation of the imputation accuracy will be provided if specified.

Value

A list of imputations

Author(s)

Hsiao-Chi Liao and InfinityFlow (Becht et. al, 2021)

`initM` *Initial biological clusters*

Description

Generating the M matrix for removing well effects.

Usage

```
initM(paths, assay, bkb.v, plots = TRUE)
```

Arguments

`paths`
: a vector of characters of paths to store intput, intermediary results, outputs...

`assay`
: the type of the input data - MPC or FFC.

`bkb.v`
: a vector of the names of the backbone markers (MUST match to the names in the FCS file).

`plots`
: logical; if TRUE (default), produce a UMAP embedding plot to visualise the structure of the biological clusters to form the initial M matrix for removal of well effect.

Details

This function has been designed to find initial biological clusters with centred transformed data.

Updating the metadata for cells in the fcs_metadata_df.rds file, adding the information of the initial biological clusters, and visualising the result with the scatter plots in the output directory.

Value

Metadata for cells with the initial biological clusters labels added

Author(s)

Hsiao-Chi Liao

`MapfxFFC` *Normalising data from the Fluorescence Flow Cytometry (FFC) Experiments with mapfx.norm*

Description

This function is used to normalise, including background correction and removal of batch effects, protein intensity data from FFC assays. The input data is in FCS format. The functions include data normalisation and cluster analysis.

Usage

```
MapfxFFC(
  runVignette = FALSE,
  runVignette_meta = NULL,
  runVignette_rawInten = NULL,
  FCSpath = NULL,
  Outpath = NULL,
  protein.v = NULL,
  protein.upper.quantile = 0.9,
  protein.lower.quantile = 0.1,
  protein.min.quantile = 0.01,
  plots.bkc.protein = TRUE,
  plots.initM = TRUE,
  plots.rmBatchEffect = TRUE,
  cluster.analysis.protein = TRUE,
  plots.cluster.analysis.protein = TRUE
)
```

Arguments

`runVignette`	logical; if FALSE (default), specify a path to `FCSpath` argument; TRUE for running Vignette using built-in data.
`runVignette_meta`	the argument for the built-in metadata when running Vignette; NULL (default).
`runVignette_rawInten`	the argument for the built-in raw intensities when running Vignette; NULL (default).
`FCSpath`	path to the input directory where `filename_meta.csv` and FCS files are stored. `filename_meta.csv` should be saved under `FCSpath/FCS/meta/` and FCS files should be saved under `FCSpath/FCS/fcs/`. See Vignette for details.
`Outpath`	path to the output directory where intermediate results and final results will be stored.

`protein.v` a vector of the names of the protein markers (MUST be the same as the names in the FCS files). For example, `protein.v = c("FSC-H","FSC-W","SSC-H","SSC-W","CD3","CD4","`

`protein.upper.quantile`
the cut-off (default = 0.9) for selecting cells used for estimating the parameter of signal for protein markers.

`protein.lower.quantile`
the cut-off (default = 0.1) for selecting cells used for estimating the parameters of noise for protein markers.

`protein.min.quantile`
the cut-off (default = 0.01) for omitting the cells with the smallest values to minimise the impact of outliers during estimation.

`plots.bkc.protein`
logical; if TRUE (default), produce scatter plots for pre- and post- background adjusted protein markers (calibrated values on y-axis and raw values on x-axis).

`plots.initM` logical; if TRUE (default), produce an UMAP embedding plot to visualise the structure of the biological clusters used to form the initial M matrix for removal of batch effects.

`plots.rmBatchEffect`
logical; if TRUE (default), produce heatmaps to visualise the unwanted (batch) effects and biological effects in the pre- and post- adjusted datasets.

`cluster.analysis.protein`
logical; if TRUE (default), perform cluster analysis using normalised protein markers.

`plots.cluster.analysis.protein`
logical; if TRUE (default), produce an UMAP embedding plot from the normalised protein markers to visualise the structure of the biological clusters.

Details

In the output directory, this function produces the normalised protein measurements, cell group labels from the cluster analysis using normalised proteins, and graphs will be provided if specified.

Value

Normalised protein markers on log scale and metadata for cells

Author(s)

Hsiao-Chi Liao, Agus Salim

Examples

```
# import built-in data
data(ord.fcs.raw.meta.df.out_ffc)
data(ord.fcs.raw.mt_ffc)

# create an Output directory for the MapfxFFC function
dir.create(file.path(tempdir(), "FFCnorm_Output"))

MapfxFFC_obj <- MapfxFFC(
  runVignette = TRUE, #set FALSE if not running this example
  runVignette_meta = ord.fcs.raw.meta.df.out_ffc, #set NULL if not running this example
  runVignette_rawInten = ord.fcs.raw.mt_ffc, #set NULL if not running this example
```

```
FCSpath = NULL, # users specify their own input path if not running this example
Outpath = file.path(tempdir(), "FFCnorm_Output"),
protein.v = c("CD3","CD4","CD8","CD45"),
protein.upper.quantile = 0.9,
protein.lower.quantile = 0.1,
protein.min.quantile = 0.01,
plots.bkc.protein = TRUE,
plots.initM = TRUE,
plots.rmBatchEffect = TRUE,
cluster.analysis.protein = TRUE, plots.cluster.analysis.protein = TRUE)
```

MapfxMPC *MAssively Parallel Flow cytometry Xplorer (MAPFX)*

Description

This function is an end-to-end toolbox for analysing single-cell protein intensity data from the Massively-Parallel Cytometry (MPC) Experiments in FCS format. The functions include data normalisation, imputation (using backbone markers), and cluster analysis.

Usage

```
MapfxMPC(
  runVignette = FALSE,
  runVignette_meta = NULL,
  runVignette_rawInten = NULL,
  FCSpath = NULL,
  Outpath = NULL,
  file_meta = "auto",
  bkb.v = NULL,
  yvar = "Legend",
  control.wells = NULL,
  bkb.upper.quantile = 0.9,
  bkb.lower.quantile = 0.1,
  bkb.min.quantile = 0.01,
  inf.lower.quantile = 0.1,
  inf.min.quantile = 0.01,
  plots.bkc.bkb = TRUE,
  plots.bkc.inf = TRUE,
  plots.initM = TRUE,
  plots.rmWellEffect = TRUE,
  impute = TRUE,
  models.use = c("XGBoost"),
  extra_args_regression_params = list(list(nrounds = 1500, eta = 0.03)),
  prediction_events_downsampling = NULL,
  impu.training = FALSE,
  plots.imputation = TRUE,
  cluster.analysis.bkb = TRUE,
  plots.cluster.analysis.bkb = TRUE,
  cluster.analysis.all = TRUE,
```

```
  plots.cluster.analysis.all = TRUE,
  cores = 4L
)
```

Arguments

runVignette	logical; if FALSE (default), specify a path to FCSpath argument; TRUE for running Vignette using built-in data.
runVignette_meta	the argument for the built-in metadata when running Vignette; NULL (default).
runVignette_rawInten	the argument for the built-in raw intensities when running Vignette; NULL (default).
FCSpath	path to the input directory where filename_meta.csv and FCS files are stored (one file per well). filename_meta.csv should be saved under FCSpath/FCS/meta/ and FCS files should be saved under FCSpath/FCS/fcs/ (See Vignette for details.)
Outpath	path to the output directory where intermediate results and final results will be stored.
file_meta	if the file names of the FCS files are in the specified format, set file_meta = "auto"; otherwise set file_meta = "usr" and provide a filename_meta.csv file in FCSpath/FCS/meta/.
bkb.v	a vector of the names of the backbone markers (MUST be the same as the names in the FCS files). For example, bkb.v = c("FSC-H", "FSC-W", "SSC-H", "SSC-W", "CD69-CD301b", "MHCII", "CD4", "CD44", "CD8", "CD11c", "CD11b", "F480", "Ly6C", "Lineage", "CD45a488", "CD24", "CD103").
yvar	the name of the well-specific exploratory marker in the FCS files (e.g., "Legend").
control.wells	the well label of the control wells, including the autofluorescence and the isotype controls (format: plate_well, e.g., P1_A01). Users need to provide this information when cluster.analysis.all = TRUE. For example, control.wells = c("P1_A01", "P2_A01", "P3_A01", "P3_F04", "P3_F05", "P3_F06", "P3_F07", "P3_F08", "P3_F09", "P3_F10", "P3_F11", "P3_F12", "P3_G01", "P3_G02").
bkb.upper.quantile	the cut-off (default = 0.9) for selecting cells used for estimating the parameter of signal for backbone markers.
bkb.lower.quantile	the cut-off (default = 0.1) for selecting cells used for estimating the parameters of noise for backbone markers.
bkb.min.quantile	the cut-off (default = 0.01) for omitting the cells with the smallest values to minimise the impact of outliers during estimation (backbone).
inf.lower.quantile	the cut-off (default = 0.1) for selecting cells used for estimating the parameters of noise for infinity markers.
inf.min.quantile	the cut-off (default = 0.01) for omitting the cells with the smallest values to minimise the impact of outliers during estimation (infinity).

`plots.bkc.bkb` logical; if TRUE (default), produce scatter plots for pre- and post- background adjusted backbone markers (calibrated values on y-axis and raw values on x-axis).

`plots.bkc.inf` logical; if TRUE (default), produce scatter plots for pre- and post- background adjusted infinity markers (calibrated values on y-axis and raw values on x-axis).

`plots.initM` logical; if TRUE (default), produce an UMAP embedding plot to visualise the structure of the biological clusters used to form the initial M matrix for removal of well effects.

`plots.rmWellEffect` logical; if TRUE (default), produce heatmaps to visualise the unwanted (well) effects and biological effects in the pre- and post- adjusted datasets.

`impute` logical; if TRUE (default), impute the missing infinity markers.

`models.use` a vector of the names of the models used for imputation. For example, `models.use = c("LM", "LASSO3", "SVM", "XGBoost")`.

`extra_args_regression_params` a list of the lists of the parameters for running regression models. The order should be the same as the models specified in `models.use`. For example, `extra_args_regression_params = list(list(degree = 1), list(nfolds = 10, degree = 3), list(type = "nu-regression", cost = 8, nu = 0.5, kernel = "radial"), list(nrounds = 1500, eta = 0.03))`.

`prediction_events_downsampling` integer (default = NULL); the number of samples used for the downsampling for the prediction.

`impu.training` logical; if FALSE (default), not impute the training set (the dataset used to train the imputation models).

`plots.imputation` logical; if TRUE (default), visualise the distribution of R-sq values of infinity markers.

`cluster.analysis.bkb` logical; if TRUE (default), perform cluster analysis using normalised backbone markers for all cells.

`plots.cluster.analysis.bkb` logical; if TRUE (default), produce an UMAP embedding plot from the normalised backbone markers to visualise the structure of the biological clusters for all cells.

`cluster.analysis.all` logical; must set `FALSE` if `impute = FALSE`; if TRUE (default), perform cluster analysis using normalised backbone markers and imputed infinity markers for cells in testing set.

`plots.cluster.analysis.all` logical; must set `FALSE` if `impute = FALSE`; if TRUE (default), produce an UMAP embedding plot from the normalised backbone markers and the imputed infinity markers to visualise the structure of the biological clusters for cells in testing set.

`cores` the number of cores used to perform parallel computation during the imputation process (default = 4L).

Details

In the output directory, this function produces the normalised backbone measurements, the background corrected infinity measurements, and imputed infinity markers (if set impute = TRUE), cell

group labels from the cluster analysis using both normalised backbones and the completed dataset (if impute = TRUE), and graphs will be provided if specified.

Value

Normalised backbone markers on log scale, background noise corrected infinity markers, imputations, and metadata for cells

Author(s)

Hsiao-Chi Liao, Agus Salim, and InfinityFlow (Becht et. al, 2021)

Examples

```
# import built-in data
data(ord.fcs.raw.meta.df.out_mpc)
data(ord.fcs.raw.mt_mpc)

# create an Output directory for the MapfxMPC function
dir.create(file.path(tempdir(), "MPC_impu_Output"))

# When `impute = TRUE`, randomly selecting 50% of the cells in each well for model training
set.seed(123)
MapfxMPC_impu_obj <- MapfxMPC(
  runVignette = TRUE, #set FALSE if not running this example
  runVignette_meta = ord.fcs.raw.meta.df.out_mpc, #set NULL if not running this example
  runVignette_rawInten = ord.fcs.raw.mt_mpc, #set NULL if not running this example
  FCSpath = NULL, # users specify their own input path if not running this example
  Outpath = file.path(tempdir(), "MPC_impu_Output"),
  file_meta = "auto",
  bkb.v = c(
    "FSC-H", "FSC-W", "SSC-H", "SSC-W", "CD69-CD301b", "MHCII",
    "CD4", "CD44", "CD8", "CD11c", "CD11b", "F480",
    "Ly6C", "Lineage", "CD45a488", "CD24", "CD103"),
  yvar = "Legend",
  control.wells = c(
    "P1_A01", "P2_A01", "P3_A01",
    "P3_F04", "P3_F05", "P3_F06", "P3_F07", "P3_F08",
    "P3_F09", "P3_F10", "P3_F11", "P3_F12",
    "P3_G01", "P3_G02"),
  bkb.upper.quantile = 0.9,
  bkb.lower.quantile = 0.1,
  bkb.min.quantile = 0.01,
  inf.lower.quantile = 0.1,
  inf.min.quantile = 0.01,
  plots.bkc.bkb = TRUE, plots.bkc.inf = TRUE,
  plots.initM = TRUE,
  plots.rmWellEffect = TRUE,
  impute = TRUE,
  models.use = c("XGBoost"),
  extra_args_regression_params = list(list(nrounds = 1500, eta = 0.03)),
  prediction_events_downsampling = NULL,
  impu.training = FALSE,
  plots.imputation = TRUE,
  cluster.analysis.bkb = TRUE, plots.cluster.analysis.bkb = TRUE,
  cluster.analysis.all = TRUE, plots.cluster.analysis.all = TRUE,
  cores = 2L)
```

mkImputeMT *Making the input for imputation*

Description

This function has been designed to combine the normalised backbone measurements and the normalised PE markers for later imputation.

Usage

```
mkImputeMT(paths)
```

Arguments

paths	a vector of characters of paths to store intput, intermediary results, outputs...

Details

Generating the combined data and saving to impu.input_log.mt.rds (on log scale) in the output directory.

Value

Combined normalised backbone and infinity markers for imputation

Author(s)

Hsiao-Chi Liao

ord.fcs.raw.meta.df.out_ffc *Metadata of the example MPC data*

Description

Metadata of the example MPC data

Usage

```
data(ord.fcs.raw.meta.df.out_ffc)
```

Format

a data.frame

`ord.fcs.raw.meta.df.out_mpc` *Metadata of the example FFC data*

Description

Metadata of the example FFC data

Usage

```
data(ord.fcs.raw.meta.df.out_mpc)
```

Format

a data.frame

`ord.fcs.raw.mt_ffc` *Subset of the single-cell murine lung data at steady state from an MPC experiment*

Description

Subset of the single-cell murine lung data at steady state from an MPC experiment

Usage

```
data(ord.fcs.raw.mt_ffc)
```

Format

a matrix containing cells from 266 wells (50 cells/well)

Source

https://flowrepository.org/id/FR-FCM-Z2LP

ord.fcs.raw.mt_mpc *Subset of the sorted CD4+ and CD8+ T cells from mice splenocytes from an FFC experiment*

Description

Subset of the sorted CD4+ and CD8+ T cells from mice splenocytes from an FFC experiment

Usage

```
data(ord.fcs.raw.mt_mpc)
```

Format

a matrix containing cells from 5 batches (50 cells/batch)

Source

http://flowrepository.org/id/FR-FCM-Z6UG

rmBatchEffect *Removing batch effect from the data (FFC)*

Description

This function has been designed to remove the unwanted effects (batch effects) from the background corrected measurements.

Usage

```
rmBatchEffect(paths, plots = TRUE)
```

Arguments

paths	a vector of characters of paths to store intput, intermediary results, outputs...
plots	logical; if TRUE (default), produce heatmaps to visualise the unwanted (batch) effects and biological effects in the pre- and post- adjusted datasets.

Details

Generating the calibrated measurements and saving to bkc.adj.bkb_logScale_mt.rds (on log scale) and bkc.adj.bkb_linearScale_mt.rds (on linear scale), and visualising the result with the heatmaps in the output directory.

Value

Normalised markers on log scale

Author(s)

Hsiao-Chi Liao

`rmWellEffect` *Removing well effect from the data*

Description

This function has been designed to remove the unwanted effects (well effects) from the background corrected measurements.

Usage

```
rmWellEffect(paths, plots = TRUE)
```

Arguments

`paths`	a vector of characters of paths to store intput, intermediary results, outputs...
`plots`	logical; if TRUE (default), produce heatmaps to visualise the unwanted (well) effects and biological effects in the pre- and post- adjusted datasets.

Details

Generating the calibrated measurements and saving to bkc.adj.bkb_logScale_mt.rds (on log scale) and bkc.adj.bkb_linearScale_mt.rds (on linear scale), and visualising the result with the heatmaps in the output directory.

Value

Normalised backbone markers on log scale

Author(s)

Hsiao-Chi Liao

Index