

Package ‘CATALYST’

April 15, 2019

Type Package

Title Cytometry dATa anALYSis Tools

Version 1.6.7

Depends R (>= 3.5)

biocViews DifferentialExpression, FlowCytometry, ImmunoOncology, MassSpectrometry, Normalization, Preprocessing, SingleCell, Software, StatisticalMethod, Visualization

Description Mass cytometry (CyTOF) uses heavy metal isotopes rather than fluorescent tags as reporters to label antibodies, thereby substantially decreasing spectral overlap and allowing for examination of over 50 parameters at the single cell level. While spectral overlap is significantly less pronounced in CyTOF than flow cytometry, spillover due to detection sensitivity, isotopic impurities, and oxide formation can impede data interpretability. We designed CATALYST (Cytometry dATa anALYSis Tools) to provide a pipeline for preprocessing of cytometry data, including i) normalization using bead standards, ii) single-cell deconvolution, and iii) bead-based compensation.

Imports Biobase, circlize, ComplexHeatmap, ConsensusClusterPlus, data.table, dplyr, drc, DT, flowCore, FlowSOM, ggplot2, ggrepel, ggridges, graphics, grDevices, grid, gridExtra, htmltools, limma, magrittr, matrixStats, methods, nnls, plotly, purrr, RColorBrewer, reshape2, Rtsne, S4Vectors, scales, scatter, shiny, shinydashboard, shinyBS, shinyjs, SingleCellExperiment, stats, SummarizedExperiment, utils

Suggests BiocStyle, knitr, rmarkdown, testthat, diffcyt, umap

URL <https://github.com/HelenaLC/CATALYST>

BugReports <https://github.com/HelenaLC/CATALYST/issues>

VignetteBuilder knitr

RoxygenNote 6.1.1

License GPL (>=2)

LazyData TRUE

Encoding UTF-8

git_url <https://git.bioconductor.org/packages/CATALYST>

git_branch RELEASE_3_8

git_last_commit 88805f2

git_last_commit_date 2019-03-21

Date/Publication 2019-04-15

Author Helena L. Crowell [aut, cre],
 Vito R.T. Zanotelli [aut],
 Stéphane Chevrier [aut, dtc],
 Mark D. Robinson [aut, fnd],
 Bernd Bodenmiller [fnd]

Maintainer Helena L. Crowell <helena.crowell@uzh.ch>

R topics documented:

adaptSpillmat	3
applyCutoffs	4
assignPrelim	5
cluster	6
compCytof	7
computeSpillmat	9
concatFCS	10
daFrame-class	12
data	13
dbFrame-class	15
dbFrame-methods	16
ei	18
estCutoffs	20
extractClusters	22
filter	23
guessPanel	24
launchGUI	25
mergeClusters	25
normCytof	26
outFCS	28
outFrames	29
plotAbundances	30
plotClusterExprs	31
plotClusterHeatmap	32
plotCodes	33
plotCounts	34
plotDiffHeatmap	35
plotDR	37
plotEvents	38
plotExprHeatmap	39
plotExprs	40
plotMahal	41
plotMDS	42
plotMedExprs	43
plotNRS	44
plotSpillmat	45
plotYields	46
runDR	47

Index

50

adaptSpillmat	<i>Adapt spillover matrix</i>
---------------	-------------------------------

Description

This helper function adapts the columns of a provided spillover matrix such that it is compatible with data having the column names provided.

Usage

```
adaptSpillmat(input_sm, out_chs, ...)  
  
## S4 method for signature 'matrix,vector'  
adaptSpillmat(input_sm, out_chs,  
  isotope_list = CATALYST::isotope_list)
```

Arguments

input_sm	a previously calculated spillover matrix.
out_chs	the column names that the prepared output spillover matrix should have. Numeric names as well as names of the form MetalMass(Di), e.g. Ir191Di or Ir191, will be interpreted as masses with associated metals.
...	optional arguments.
isotope_list	named list. Used to validate the input spillover matrix. Names should be metals; list elements numeric vectors of their isotopes. See isotope_list for the list of isotopes used by default.

Details

The rules how the spillover matrix is adapted are explained in [compCytof](#).

Value

An adapted spillover matrix with column and row names according to out_chs.

Author(s)

Helena Lucia Crowell <helena.crowell@uzh.ch> and Vito RT Zanutelli <vito.zanutelli@uzh.ch>

Examples

```
# get single-stained control samples  
data(ss_exp)  
# specify mass channels stained for  
bc_ms <- c(139, 141:156, 158:176)  
# debarcode  
re <- assignPrelim(x = ss_exp, y = bc_ms)  
re <- estCutoffs(x = re)  
re <- applyCutoffs(x = re)  
# estimate spillover matrix and adapt it  
sm <- computeSpillmat(x = re)  
chs <- flowCore::colnames(ss_exp)
```

```
adaptSpillmat(sm, chs)
```

```
applyCutoffs          Single-cell debarcoding (2)
```

Description

Applies separation and mahalanobies distance cutoffs.

Usage

```
applyCutoffs(x, ...)

## S4 method for signature 'dbFrame'
applyCutoffs(x, mhl_cutoff = 30,
             sep_cutoffs = NULL)
```

Arguments

x	a dbFrame .
...	optional arguments.
mhl_cutoff	mahalanobis distance threshold above which events should be unassigned. This argument will be ignored if the mhl_cutoff slot of the input dbFrame is specified.
sep_cutoffs	non-negative numeric of length one or of same length as the number of rows in the bc_key(x). Specifies the distance separation cutoffs between positive and negative barcode populations below which events should be unassigned. If NULL (default), applyCutoffs will try to access the sep_cutoffs slot of the input dbFrame.

Value

Will update the bc_ids and, if not already specified, sep_cutoffs & mhl_cutoff slots of x.

Author(s)

Helena Lucia Crowell <helena.crowell@uzh.ch>

References

Zunder, E.R. et al. (2015). Palladium-based mass tag cell barcoding with a doublet-filtering scheme and single-cell deconvolution algorithm. *Nature Protocols* **10**, 316-333.

Examples

```

data(sample_ff, sample_key)
re <- assignPrelim(x = sample_ff, y = sample_key)

# use global separation cutoff
applyCutoffs(x = re, sep_cutoffs = 0.4)

# estimate population-specific cutoffs
re <- estCutoffs(x = re)
applyCutoffs(x = re)

```

assignPrelim	<i>Single-cell debarcoding (1)</i>
--------------	------------------------------------

Description

Assigns a preliminary barcode ID to each event.

Usage

```

assignPrelim(x, y, ...)

## S4 method for signature 'flowFrame,data.frame'
assignPrelim(x, y, cofactor = 10,
             verbose = TRUE)

## S4 method for signature 'flowFrame,vector'
assignPrelim(x, y, cofactor = 10,
             verbose = TRUE)

## S4 method for signature 'character,data.frame'
assignPrelim(x, y, cofactor = 10,
             verbose = TRUE)

## S4 method for signature 'character,vector'
assignPrelim(x, y, cofactor = 10,
             verbose = TRUE)

```

Arguments

x	a flowFrame or character of an FCS file name.
y	the debarcoding scheme. A binary matrix with sample names as row names and numeric masses as column names OR a vector of numeric masses corresponding to barcode channels. When the latter is supplied, ‘assignPrelim’ will create a scheme of the appropriate format internally.
...	optional arguments.
cofactor	numeric. Cofactor used for asinh transformation.
verbose	logical. Should extra information on progress be reported?

Value

Returns a [dbFrame](#) containing measurement intensities, the debarcoding key, a numeric vector of barcode IDs and separations between positive and negative barcode populations, and barcode intensities normalized by population.

Author(s)

Helena Lucia Crowell <helena.crowell@uzh.ch>

References

Zunder, E.R. et al. (2015). Palladium-based mass tag cell barcoding with a doublet-filtering scheme and single-cell deconvolution algorithm. *Nature Protocols* **10**, 316-333.

Examples

```
data(sample_ff, sample_key)
assignPrelim(x = sample_ff, y = sample_key)
```

cluster

FlowSOM clustering & ConsensusClusterPlus metaclustering

Description

cluster will first group cells into x dim y dim clusters using **FlowSOM**, and subsequently perform metaclustering with **ConsensusClusterPlus** into 2 through maxK clusters. In the returned daFrame, those antigens used for clustering will be labelled as 'type' markers, and the remainder of antigens as 'state' markers.

Usage

```
cluster(x, ...)

## S4 method for signature 'daFrame'
cluster(x, cols_to_use = NULL, xdim = 10,
        ydim = 10, maxK = 20, verbose = TRUE, seed = 1)
```

Arguments

x	a daFrame .
...	optional arguments.
cols_to_use	a character vector. Specifies which antigens to use for clustering. The default (NULL) uses type_markers(x). Must be provided if colData(x)\$marker_class has not been specified.
xdim, ydim	numeric. Specify the grid size of the self-organizing map. The default 10x10 grid will yield 100 clusters.
maxK	numeric. Specifies the maximum number of clusters to evaluate in the metaclustering. For maxK = 20, for example, metaclustering will be performed for 2 through 20 clusters.
verbose	logical. Should information on progress be reported?
seed	numeric. Sets random seed in ConsensusClusterPlus().

Details

The delta area represents the amount of extra cluster stability gained when clustering into k groups as compared to $k-1$ groups. It can be expected that high stability of clusters can be reached when clustering into the number of groups that best fits the data. The "natural" number of clusters present in the data should thus correspond to the value of k where there is no longer a considerable increase in stability (plateau onset).

Value

The function will add information to the following slots of the input `daFrame` (and return it):

- `rowData`
 - `cluster_id`: each cell's cluster ID as inferred by FlowSOM. One of 1, ..., $x\text{dim} \times y\text{dim}$.
- `colData`
 - `marker_class`: "type" or "state". Specifies whether an antigen has been used for clustering or not, respectively.
- `metadata`
 - `SOM_codes`: a table with dimensions $K \times (\# \text{ cell type markers})$, where $K = x\text{dim} \times y\text{dim}$. Contains the SOM codes.
 - `cluster_codes`: a table with dimensions $K \times (\text{maxK} + 1)$. Contains the cluster codes for all metaclustering.
 - `delta_area`: a `ggplot` object. See above for details.

Author(s)

Helena Lucia Crowell <helena.crowell@uzh.ch>

References

Nowicka M, Krieg C, Weber LM et al. CyTOF workflow: Differential discovery in high-throughput high-dimensional cytometry datasets. *F1000Research* 2017, 6:748 (doi: 10.12688/f1000research.11622.1)

Examples

```
data(PBMC_fs, PBMC_panel, PBMC_md)
daf <- daFrame(PBMC_fs, PBMC_panel, PBMC_md)

# run clustering
(daf <- cluster(daf))
```

compCytof

Compensate CyTOF data

Description

Compensates a mass spectrometry based experiment using a provided spillover matrix & assuming a linear spillover in the experiment.

Usage

```

compCytof(x, y, ...)

## S4 method for signature 'flowFrame,matrix'
compCytof(x, y, out_path = NULL,
  method = "flow", isotope_list = CATALYST::isotope_list)

## S4 method for signature 'flowSet,ANY'
compCytof(x, y, out_path = NULL,
  method = "flow")

## S4 method for signature 'character,matrix'
compCytof(x, y, out_path = NULL,
  method = "flow")

## S4 method for signature 'ANY,data.frame'
compCytof(x, y, out_path = NULL,
  method = "flow")

```

Arguments

x	a flowFrame OR a character string specifying the location of FCS files that should be compensated.
y	a spillover matrix.
...	optional arguments.
out_path	a character string. If specified, compensated FCS files will be generated in this location. If x is a character string, file names will be inherited from uncompensated FCS files and given extension "_comped".
method	"flow" or "nnls".
isotope_list	named list. Used to validate the input spillover matrix. Names should be metals; list elements numeric vectors of their isotopes. See isotope_list for the list of isotopes used by default.

Details

If the spillover matrix (SM) does not contain the same set of columns as the input experiment, it will be adapted according to the following rules:

1. columns present in the SM but not in the input data will be removed from it
2. non-metal columns present in the input but not in the SM will be added such that they do neither receive nor cause spill
3. metal columns that have the same mass as a channel present in the SM will receive (but not emit) spillover according to that channel
4. if an added channel could potentially receive spillover (as it has +/-1M or +16M of, or is of the same metal type as another channel measured), a warning will be issued as there could be spillover interactions that have been missed and may lead to faulty compensation

Value

Compensates the input [flowFrame](#) or, if x is a character string, all FCS files in the specified location. If out_path=NULL (the default), returns a [flowFrame](#) containing the compensated data. Otherwise, compensated data will be written to the specified location as FCS 3.0 standard files.

Author(s)

Helena Lucia Crowell <helena.crowell@uzh.ch> and Vito Zanutelli <vito.zanutelli@uzh.ch>

Examples

```
# get single-stained control samples
data(ss_exp)

# specify mass channels stained for
bc_ms <- c(139, 141:156, 158:176)

# debarcode
re <- assignPrelim(x = ss_exp, y = bc_ms)
re <- estCutoffs(x = re)
re <- applyCutoffs(x = re)
spillMat <- computeSpillmat(x = re)
compCytotof(x = ss_exp, y = spillMat)
```

computeSpillmat	<i>Compute spillover matrix</i>
-----------------	---------------------------------

Description

Computes a spillover matrix from a priori identified single-positive populations.

Usage

```
computeSpillmat(x, ...)

## S4 method for signature 'dbFrame'
computeSpillmat(x, method = "default",
  interactions = "default", trim = 0.5, th = 1e-05)
```

Arguments

x	a dbFrame .
...	optional arguments.
method	"default" or "classic". Specifies the function to be used for spillover estimation (see below for details).
interactions	"default" or "all". Specifies which interactions spillover should be estimated for. The default exclusively takes into consideration interactions that are sensible from a chemical and physical point of view (see below for more details).
trim	numeric. Specifies the trim value used for estimation of spill values. Note that trim = 0.5 is equivalent to using medians.
th	single non-negative numeric. Specifies the threshold value below which spill estimates will be set to 0.

Details

The default method estimates the spillover as the median ratio between the unstained spillover receiving and the stained spillover emitting channel in the corresponding single stained populations.

method = "classic" will compute the slope of a line through the medians (or trimmed means) of stained and unstained populations. The medians (or trimmed means) computed from events that are i) negative in the respective channels; and, ii) not assigned to interacting channels; and, iii) not unassigned are subtracted as to account for background.

interactions="default" considers only expected interactions, that is, M+/-1 (detection sensitivity), M+16 (oxide formation) and channels measuring metals that are potentially contaminated by isotopic impurities (see reference below and [isotope_list](#)).

interaction="all" will estimate spill for all $n \times n - n$ interactions, where n denotes the number of single-color controls (= nrow(bc_key(re))).

Value

Returns a square compensation matrix with dimensions and dimension names matching those of the input flowFrame. Spillover is assumed to be linear, and, on the basis of their additive nature, spillover values are computed independently for each interacting pair of channels.

Author(s)

Helena Lucia Crowell <helena.crowell@uzh.ch>

References

Coursey, J.S., Schwab, D.J., Tsai, J.J., Dragoset, R.A. (2015). Atomic weights and isotopic compositions, (available at <http://physics.nist.gov/Comp>).

Examples

```
# get single-stained control samples
data(ss_exp)
# specify mass channels stained for
bc_ms <- c(139, 141:156, 158:176)
# debarcode single-positive populations
re <- assignPrelim(x = ss_exp, y = bc_ms)
re <- estCutoffs(x = re)
re <- applyCutoffs(x = re)
head(computeSpillmat(x = re))
```

concatFCS

FCS file Concatenation

Description

Concatenates all input data to a single file or object.

Usage

```
concatFCS(x, ...)  
  
## S4 method for signature 'flowSet'  
concatFCS(x, out_path = NULL, fn = NULL,  
  fn_sep = "_", by_time = TRUE, file_num = FALSE, pars = NULL,  
  desc = NULL)  
  
## S4 method for signature 'character'  
concatFCS(x, out_path = NULL, fn = NULL,  
  fn_sep = "_", by_time = TRUE, file_num = FALSE)  
  
## S4 method for signature 'list'  
concatFCS(x, out_path = NULL, fn = NULL,  
  fn_sep = "_", by_time = TRUE, file_num = FALSE)
```

Arguments

x	can be either a flowSet, a list of flowFrames, a character specifying the location of the FCS files to be concatenated, or a vector of FCS file names.
...	optional arguments.
out_path	character string. If specified, an FCS file of the concatenated data will be written to this location. If NULL (default), a flowFrame will be returned.
fn	a character string to use as the output file name. Defaults to the file name of the first input FCS file or flowFrame, respectively.
fn_sep	a character string to use to separate the output file name's prefix from the appendage.
by_time	logical. Specifies whether files should be ordered by time of acquisition.
file_num	logical. Specifies whether a file number column should be added.
pars, desc	optional character vectors of channel names & descriptions to use when merging files.

Value

a flowFrame containing measurement intensities of all input data or a character of the FCS file name.

Author(s)

Helena Lucia Crowell <helena.crowell@uzh.ch>

Examples

```
data(raw_data)  
concatFCS(raw_data)
```

daFrame-class *Differential analysis frame class*

Description

Represents the data returned by and used throughout differential analysis.

assays a list of length 1 containing the measurement data.

rowData an object of the [DataFrame-class](#) containing the row metadata.

colData an object of the [DataFrame-class](#) containing the column metadata.

metadata a named list containing:

- experimental_design: the original metadata table.
- n_cells: number of events (cells) measured per sample.
- cofactor: numeric cofactor used for arcsinh-transformation.
- cluster_codes: cluster codes for the initial **FlowSOM** clustering ("som100"), the **ConsensusClusterPlus** metaclustering ("metaX"), and all manual mergings done with [mergeClusters](#).
- delta_area: the Delta Area plot (see **ConsensusClusterPlus** for details).
- experimental_design: the original metadata table

Usage

```
daFrame(x, panel, md, ...)
```

```
## S4 method for signature 'flowSet,data.frame,data.frame'
```

```
daFrame(x, panel, md,
  cols_to_use = NULL, cofactor = 5, panel_cols = list(channel =
  "fcs_colname", antigen = "antigen", class = "marker_class"),
  md_cols = list(file = "file_name", id = "sample_id", factors =
  c("condition", "patient_id")))
```

```
## S4 method for signature 'character,data.frame,data.frame'
```

```
daFrame(x, panel, md, ...)
```

```
## S4 method for signature 'daFrame,matrix,ANY'
```

```
daFrame(x, panel, md, ...)
```

```
## S4 method for signature 'daFrame,ANY,matrix'
```

```
daFrame(x, panel, md, ...)
```

Arguments

x a flowSet holding all samples or a path to a set of FCS files.

panel a data.frame containing, for each channel, its column name in the input data, targeted protein marker, and (optionally) class ("type", "state", or "none").

md a table with column describing the experiment. An exemplary metadata table could look as follows:

- file_name: the FCS file name
- sample_id: a unique sample identifier

	<ul style="list-style-type: none"> • <code>patient_id</code>: the patient ID • <code>condition</code>: brief sample description (e.g. reference/stimulated, healthy/diseased)
<code>...</code>	optional arguments.
<code>cols_to_use</code>	a logical vector, numeric vector of column indices, or character vector of channel names. Specified which column to keep from the input data. Defaults to the channels listed in the input panel.
<code>cofactor</code>	numeric cofactor to use for arcsinh-transformation.
<code>panel_cols</code>	a names list specifying the column names of <code>panel</code> that contain the channel names, targeted protein markers, and (optionally) marker classes.
<code>md_cols</code>	a named list specifying the column names of <code>md</code> that contain the FCS file names, sample IDs, and factors of interest (batch, condition, treatment etc.).

Value

an object of class `daFrame`.

Author(s)

Helena Lucia Crowell <helena.crowell@uzh.ch>

Examples

```
data(PBMC_fs, PBMC_panel, PBMC_md)
daFrame(PBMC_fs, PBMC_panel, PBMC_md)
```

data

Example data sets

Description

- Concatenation & Normalization

`raw_data` a `flowSet` with 3 experiments, each containing 2'500 raw measurements with a variation of signal over time. Samples were mixed with DVS beads capture by mass channels 140, 151, 153, 165 and 175.

- Debarcoding

`sample_ff` a `flowFrame` following a 6-choose-3 barcoding scheme where mass channels 102, 104, 105, 106, 108, and 110 were used for labeling such that each of the 20 individual barcodes are positive for exactly 3 out of the 6 barcode channels.

`sample_key` a `data.frame` of dimension 20 x 6 with sample names as row and barcode masses as column names. Contains a binary code of length 6 for each sample in `sample_ff`, e.g. 111000, as its unique identifier.

- Compensation

`ss_exp` a `flowFrame` with 20'000 events. Contains 36 single-antibody stained controls where beads were stained with antibodies captured by mass channels 139, 141 through 156, and 158 through 176, respectively, and pooled together.

`mp_cells` a `flowFrame` with 5000 spill-affected multiplexed cells and 39 measurement parameters.

isotope_list a named list of isotopic compositions for all elements within 75 through 209 u corresponding to the CyTOF mass range at the time of writing.

- Differential Analysis

PBMC_fs a `flowSet` with PBMCs samples from 6 patients. For each sample, the expression of 10 cell surface and 14 signaling markers was measured before (REF) and upon BCR/FcR-XL stimulation (BCRXL) with B cell receptor/ Fc receptor crosslinking for 30', resulting in a total of 12 samples.

PBMC_panel a 2 column data.frame that contains each marker's column name in the FCS file, and its targeted protein marker.

PBMC_md a data.frame where each row corresponds to a sample, and with columns describing the experimental design.

merging_table a 20 x 2 table with "old_cluster" IDs and "new_cluster" labels to exemplify manual cluster merging and cluster annotation.

Value

see descriptions above.

Author(s)

Helena Lucia Crowell <helena.crowell@uzh.ch>

References

Bodenmiller, B., Zunder, E.R., Finck, R., et al. (2012). Multiplexed mass cytometry profiling of cellular states perturbed by small-molecule regulators. *Nature Biotechnology* **30**(9): 858-67.

Coursey, J.S., Schwab, D.J., Tsai, J.J., Dragoset, R.A. (2015). Atomic weights and isotopic compositions, (available at <http://physics.nist.gov/Comp>).

Examples

```
### example data for concatenation & normalization:
# raw measurement data
data(raw_data)

### example data for debarcoding:
# 20 barcoded samples
data(sample_ff)
# 6-choose-3 barcoding scheme
data(sample_key)

### example data for compensation:
# single-stained control samples
data(ss_exp)
# multiplexed cells
data(mp_cells)

### example data for differential analysis:
# REF vs. BCRXL samples
data(PBMC_fs)
# antigen panel & experimental design
data(PBMC_panel, PBMC_md)
# exemplary manual merging table
data(merging_table)
```

dbFrame-class	<i>Debarcoding frame class</i>
---------------	--------------------------------

Description

This class represents the data returned by and used throughout debarcoding.

Details

Objects of class dbFrame hold all data required for debarcoding:

1. as the initial step of single-cell deconvolution, `assignPrelim` will return a dbFrame containing the input measurement data, barcoding scheme, and preliminary assignments.
2. assignments will be made final by `applyCutoffs`. Optionally, population-specific separation cutoffs may be estimated by running `estCutoffs` prior to this.
3. `plotYields`, `plotEvents` and `plotMahal` aim to guide deconvolution parameter selection, and to give a sense of the resulting barcode assignment quality.

`show(dbFrame)` will display

- the dimensionality of the measurement data and number of barcodes
- current assignments in order of decreasing population size
- current separation cutoffs
- the mean & per-population yield that'll be achieved upon debarcoding

Slots

`exprs` a matrix containing raw intensities of the input flowFrame.

`bc_key` binary barcoding scheme with numeric masses as column names and samples names as row names OR a numeric vector of barcode masses.

`bc_ids` vector of barcode IDs. If a barcoding scheme is supplied, the respective binary code's row name, else, the mass of the respective barcode channel.

`deltas` numeric vector of separations between positive and negative barcode populations computed from normalized barcode intensities.

`normed_bcs` matrix containing normalized barcode intensities.

`mhl_dists` mahalanobis distances.

`sep_cutoffs` numeric vector of distance separation cutoffs between positive and negative barcode populations above which events will be unassigned.

`mhl_cutoff` non-negative and non-zero numeric value specifying the Mahalanobis distance below which events will be unassigned.

`counts` matrix of dimension (# barcodes)x(101) where each row contains the number of events within a barcode for which positive & negative populations are separated by a distance between in [0,0.01), ..., [0.99,1], respectively.

`yields` a matrix of dimension (# barcodes)x(101) where each row contains the percentage of events within a barcode that will be obtained after applying a separation cutoff of 0, 0.01, ..., 1, respectively.

Author(s)

Helena Lucia Crowell <helena.crowell@uzh.ch>

`dbFrame-methods`*Extraction and replacement methods for objects of class dbFrame*

Description

Methods for replacing and accessing slots in a `dbFrame`.

Usage`bc_key(x)``bc_ids(x)``deltas(x)``normed_bcs(x)``mhl_dists(x)``sep_cutoffs(x)``mhl_cutoff(x)``counts(x)``yields(x)`

```
## S4 method for signature 'dbFrame'  
exprs(object)
```

```
## S4 method for signature 'dbFrame'  
bc_key(x)
```

```
## S4 method for signature 'dbFrame'  
bc_ids(x)
```

```
## S4 method for signature 'dbFrame'  
deltas(x)
```

```
## S4 method for signature 'dbFrame'  
normed_bcs(x)
```

```
## S4 method for signature 'dbFrame'  
mhl_dists(x)
```

```
## S4 method for signature 'dbFrame'  
sep_cutoffs(x)
```

```
## S4 method for signature 'dbFrame'  
mhl_cutoff(x)
```

```
## S4 method for signature 'dbFrame'
counts(x)

## S4 method for signature 'dbFrame'
yields(x)

## S4 replacement method for signature 'dbFrame,numeric'
mhl_cutoff(x) <- value

## S4 replacement method for signature 'dbFrame,ANY'
mhl_cutoff(x) <- value

## S4 replacement method for signature 'dbFrame,numeric'
sep_cutoffs(x) <- value

## S4 replacement method for signature 'dbFrame,ANY'
sep_cutoffs(x) <- value
```

Arguments

`x`, object a [dbFrame](#).
`value` the replacement value.

Value

`exprs` extracts the raw data intensities.
`bc_key` extracts the barcoding scheme.
`bc_ids` extracts currently made event assignments.
`deltas` extracts barcode separations computed from normalized intensities. `sep_cutoffs` apply to these values (see [applyCutoffs](#)).
`normed_bcs` extracts normalized barcode intensities (see [assignPrelim](#)).
`sep_cutoffs`, `sep_cutoffs<-` extracts or replaces separation cutoffs. If option `sep_cutoffs` is not specified, these will be used by [applyCutoffs](#). Replacement value must be a non-negative numeric with length one or same length as the number of barcodes.
`mhl_cutoff`, `mhl_cutoff<-` extracts or replaces the Mahalanobis distance threshold above which events are to be unassigned. Replacement value must be a single non-negative and non-zero numeric.
`counts` extract the counts matrix (see [dbFrame](#)).
`yields` extract the yields matrix (see [dbFrame](#)).

Author(s)

Helena Lucia Crowell <helena.crowell@uzh.ch>

Examples

```
data(sample_ff, sample_key)
re <- assignPrelim(x = sample_ff, y = sample_key)

# set global cutoff parameter
sep_cutoffs(re) <- 0.4
```

```

re <- applyCutoffs(x = re)

# subset a specific population, e.g. A1: 111000
a1 <- bc_ids(re) == "A1"
head(exprs(sample_ff[a1, ]))

# subset unassigned events
unassigned <- bc_ids(re) == 0
head(exprs(sample_ff[unassigned, ]))

```

ei

*Extraction and replacement methods for objects of class daFrame***Description**

Methods for accessing slots in a [daFrame](#).

Usage

```

ei(x)

n_cells(x)

marker_classes(x)

type_markers(x)

state_markers(x)

sample_ids(x)

cluster_codes(x)

cluster_ids(x, ...)

## S4 method for signature 'daFrame'
reducedDimNames(x)

## S4 method for signature 'daFrame'
reducedDims(x)

## S4 method for signature 'daFrame'
reducedDim(x, dr = 1)

## S4 replacement method for signature 'daFrame,character'
reducedDimNames(x) <- value

## S4 method for signature 'daFrame'
exprs(object)

## S4 method for signature 'daFrame'
ei(x)

```

```

## S4 method for signature 'daFrame'
n_cells(x)

## S4 method for signature 'daFrame'
marker_classes(x)

## S4 method for signature 'daFrame'
type_markers(x)

## S4 method for signature 'daFrame'
state_markers(x)

## S4 method for signature 'daFrame'
sample_ids(x)

## S4 method for signature 'daFrame'
cluster_codes(x)

## S4 method for signature 'daFrame'
cluster_ids(x, k = NULL)

## S4 method for signature 'daFrame,ANY,ANY,ANY'
x[i, j]

```

Arguments

x, object	a daFrame .
...	optional arguments.
dr	character string specifying the dim. reduction to extract.
value	a character vector containing the desired dimensionality reduction names.
k	character string specifying the clustering to extract. Valid values are names(cluster_codes(x)).
i, j	indices specifying elements to extract.

Value

[exprs](#) extracts the arcsinh-transformed expressions.
[n_cells](#) extracts the number of events measured per sample.
[type_markers](#) extracts the antigens used for clustering.
[state_markers](#) extracts antigens that were not used for clustering.
[sample_ids](#) extracts the sample IDs as specified in the metadata-table.
[cluster_codes](#) extracts a data.frame containing cluster codes for the [FlowSOM](#) clustering, the [ConsensusClusterPlus](#) metaclustering, and all mergings done through [mergeClusters](#).
[cluster_ids](#) extracts the numeric vector of cluster IDs as inferred by [FlowSOM](#).

Author(s)

Helena Lucia Crowell <helena.crowell@uzh.ch>

Examples

```

# construct daFrame
data(PBMC_fs, PBMC_panel, PBMC_md)
re <- daFrame(PBMC_fs, PBMC_panel, PBMC_md)

# run clustering
re <- cluster(re)

# view data summary
library(SummarizedExperiment)
cbind(metadata(re)$experiment_info, cells=n_cells(re))

# access row / cell data
head(rowData(re))
plot(table(cluster_ids(re)))

# access marker information
type_markers(re)
state_markers(re)

# get cluster ID correspondece between 2 clusterings
old_ids <- seq_len(20)
m <- match(old_ids, cluster_codes(re)$`meta20`)
new_ids <- cluster_codes(re)$`meta12`[m]
data.frame(old_ids, new_ids)

# plot relative change in area under CDF curve vs. k
metadata(re)$delta_area

```

estCutoffs

Estimation of distance separation cutoffs

Description

For each sample, estimates a cutoff parameter for the distance between positive and negative barcode populations.

Usage

```
estCutoffs(x, ...)
```

```
## S4 method for signature 'dbFrame'
estCutoffs(x)
```

Arguments

x a [dbFrame](#).

... optional arguments.

Details

For the estimation of cutoff parameters, we considered yields upon debarcoding as a function of the applied cutoffs. Commonly, this function will be characterized by an initial weak decline, where doublets are excluded, and subsequent rapid decline in yields to zero. In between, low numbers of counts with intermediate barcode separation give rise to a plateau. As an adequate cutoff estimate, we target the point that approximately marks the end of the plateau regime and the onset of yield decline. To facilitate robust cutoff estimation, we fit a linear and a three-parameter log-logistic function to the yields function:

$$f(x) = \frac{d}{1 + e^{b(\log(x) - \log(e))}}$$

The goodness of the linear fit relative to the log-logistic fit is weighed with:

$$w = \frac{RSS_{\log\text{-logistic}}}{RSS_{\log\text{-logistic}} + RSS_{\text{linear}}}$$

and the cutoffs for both functions are defined as:

$$c_{\text{linear}} = -\frac{\beta_0}{2\beta_1}$$

$$c_{\log\text{-logistic}} = \operatorname{argmin}_x \left\{ \frac{|f'(x)|}{f(x)} > 0.1 \right\}$$

The final cutoff estimate is defined as the weighted mean between these estimates:

$$c = (1 - w) \cdot c_{\log\text{-logistic}} + w \cdot c_{\text{linear}}$$

Value

Will update the `sep_cutoffs` slot of the input `dbFframe` and return the latter.

Author(s)

Helena Lucia Crowell <helena.crowell@uzh.ch>

References

Finney, D.J. (1971). Probit Analysis. *Journal of Pharmaceutical Sciences* **60**, 1432.

Examples

```
data(sample_ff, sample_key)
# assign preliminary IDs
re <- assignPrelim(x = sample_ff, y = sample_key)
# estimate separation cutoffs
re <- estCutoffs(x = re)
# view exemplary estimate
plotYields(re, "A1")
```

extractClusters	<i>Extract clusters from a daFrame</i>
-----------------	--

Description

Extracts clusters from a daFrame. Populations will be either returned as a flowSet or written to FCS files, depending on argument as.

Usage

```
extractClusters(x, k, ...)

## S4 method for signature 'daFrame,character'
extractClusters(x, k, clusters = NULL,
  as = c("flowSet", "fcs"), out_dir = ".", verbose = TRUE)
```

Arguments

x	a daFrame.
k	numeric or character string. Specifies the clustering to extract populations from. Must be one of names(cluster_codes(x)).
...	optional arguments.
clusters	a character vector. Specifies which clusters to extract. NULL = all clusters.
as	"flowSet" or "fcs". Specifies whether clusters should be return as a flowSet or written to FCS files.
out_dir	a character string. Specifies where FCS files should be written to. Defaults to the working directory.
verbose	logical. Should information on progress be reported?

Value

a flowSet or character vector of the output file names.

Author(s)

Mark D Robinson and Helena Lucia Crowell <helena.crowell@uzh.ch>

Examples

```
data(PBMC_fs, PBMC_panel, PBMC_md, merging_table)
re <- daFrame(PBMC_fs, PBMC_panel, PBMC_md)

# run clustering
re <- cluster(re)

# merge clusters
re <- mergeClusters(re, k="meta20", table=merging_table, id="merging_1")
extractClusters(re, k="merging_1")
```

filter	<i>Filter daFrame</i>
--------	-----------------------

Description

Filters events/genes from a daFrame using conditional statements.

Usage

```
filter(x, ..., k)

## S4 method for signature 'daFrame,character'
filter(x, ..., k = NULL)

## S4 method for signature 'daFrame,missing'
filter(x, ..., k = NULL)
```

Arguments

x	a daFrame .
...	conditional statements separated by comma. Only rows/columns where the condition evaluates to TRUE are kept.
k	numeric or character string. Specifies the clustering to extract populations from. Must be one of <code>names(cluster_codes(x))</code> . Defaults to "som100".

Value

a daFrame.

Author(s)

Helena Lucia Crowell <helena.crowell@uzh.ch>

Examples

```
data(PBMC_fs, PBMC_panel, PBMC_md, merging_table)
daf <- daFrame(PBMC_fs, PBMC_panel, PBMC_md)

# run clustering
daf <- cluster(daf)

# one condition only, remove a single sample
filter(daf, condition == "Ref", sample_id != "Ref1")

# keep only a subset of clusters
filter(daf, cluster_id %in% c(7, 8, 18), k = "meta20")
```

`guessPanel`*Guess parameter panel*

Description

Helper function to parse information from the parameters slot of a flowFrame/flowSet.

Usage

```
guessPanel(x, ...)  
  
## S4 method for signature 'flowFrame'  
guessPanel(x)  
  
## S4 method for signature 'flowSet'  
guessPanel(x, index = 1)
```

Arguments

<code>x</code>	a flowFrame or flowSet.
<code>...</code>	optional arguments.
<code>index</code>	numeric. If x is a flowSet object, this index specifies which flowFrame to extract.

Value

a data.frame with the following columns:

- name: the parameter name as extracted from the input flowFrame,
- desc: the parameter description as extracted from the input flowFrame,
- antigen: the targeted protein markers, and
- use_channel: logical. If TRUE, the channel is expected to contain a marker and will be kept.

Author(s)

Mark D Robinson, and Helena Lucia Crowell <helena.crowell@uzh.ch>

Examples

```
# exemplary data with Time, DNA, BC channels, etc.  
data(raw_data)  
guessPanel(raw_data)
```

 launchGUI

Launch GUI

Description

Launches the CATALYST Shiny app.

Usage

```
launchGUI()
```

Details

Detailed user guides are available inside the app. To use the app online, please visit <http://imlspenticton.uzh.ch:3838/CAT>

Value

Opens a browser window with an interactive Shiny application.

Author(s)

Helena Lucia Crowell <helena.crowell@uzh.ch>

Examples

```
## Not run: launchGUI
```

 mergeClusters

Manual cluster merging

Description

mergeClusters provides a simple wrapper to store a manual merging inside the input daFrame.

Usage

```
mergeClusters(x, k, table, id)
```

```
## S4 method for signature 'daFrame,character,data.frame,character'
```

```
mergeClusters(x, k,  
  table, id)
```

Arguments

x	a daFrame .
k	a character string specifying the clustering to merge.
table	a merging table with 2 columns containing the cluster IDs to merge in the 1st, and the cluster IDs to newly assign in the 2nd column.
id	character string. Used as a label for the merging.
...	optional arguments.

Details

in the following code snippets, `x` is a `daFrame` object.

- merging codes are accesible through `cluster_codes(x)$id`
- all functions that ask for specification of a clustering (e.g. `plotAbundances`, `plotClusterHeatmap`) take the merging ID as a valid input argument.

Value

Writes the newly assignend cluster codes into the metadata slot `cluster_codes` of the input `daFrame` and returns the latter.

Author(s)

Helena Lucia Crowell <helena.crowell@uzh.ch>

References

Nowicka M, Krieg C, Weber LM et al. CyTOF workflow: Differential discovery in high-throughput high-dimensional cytometry datasets. *F1000Research* 2017, 6:748 (doi: 10.12688/f1000research.11622.1)

Examples

```
data(PBMC_fs, PBMC_panel, PBMC_md, merging_table)
re <- daFrame(PBMC_fs, PBMC_panel, PBMC_md)

# run clustering
re <- cluster(re)

# merge clusters
re <- mergeClusters(re, k="meta20", table=merging_table, id="merging")
plotClusterHeatmap(re, k="merging", hm2="pS6")
```

normCytof

Bead-based normalization

Description

an implementation of Finck et al.'s normalization of mass cytometry data using bead standards with automated bead gating.

Usage

```
normCytof(x, y, ...)

## S4 method for signature 'flowFrame'
normCytof(x, y, out_path = NULL, fn = NULL,
  fn_sep = "_", remove_beads = TRUE, norm_to = NULL, k = 500,
  trim = 5, verbose = TRUE, plot = TRUE)

## S4 method for signature 'character'
normCytof(x, y, out_path = NULL,
  remove_beads = TRUE, norm_to = NULL, k = 500, trim = 5,
  verbose = TRUE)
```

Arguments

x	a flowFrame or character of the FCS file to be normalized.
y	"dvs" (for bead masses 140, 151, 153, 165, 175) or "beta" (for bead masses 139, 141, 159, 169, 175) or a numeric vector of masses.
...	optional arguments.
out_path	a character string. If specified, outputs will be generated here. If NULL (the default), normCytof will return a flowFrame of the normalized data (if remove=FALSE) or a flowSet containing normalized cells and beads (if remove=TRUE).
fn	a character string to use as the output file name. Defaults to the file name of the input FCS file or flowFrame, respectively.
fn_sep	a character string to use to separate the output file name's prefix from the appendage.
remove_beads	logical. If TRUE (the default) beads will be removed and normalized cells and beads returned separately.
norm_to	a flowFrame or character of an FCS file from which baseline values should be computed and to which the input data should be normalized.
k	integer width of the median window used for bead smoothing.
trim	a single non-negative numeric. A <i>median +/- ... mad</i> rule is applied to the preliminary population of bead events to remove bead-bead doublets and low signal beads prior to estimating normalization factors.
verbose	logical. Should extra information on progress be reported?
plot	logical. Should bead vs. DNA scatters and beads before vs. after normalization be plotted?

Value

if out_path=NULL (the default) a [flowFrame](#) of the normalized data (if remove=FALSE) or [flowSet](#) containing normalized cells and beads (if remove=TRUE). Else, a character of the location where output FCS files and plots have been generated.

Author(s)

Helena Lucia Crowell <helena.crowell@uzh.ch>

References

Finck, R. et al. (2013). Normalization of mass cytometry data with bead standards. *Cytometry A* **83A**, 483-494.

Examples

```
data(raw_data)
ff <- concatFCS(raw_data)
normCytof(x = ff, y = "dvs", k = 120)
```

outFCS	<i>Write population-wise FCS files</i>
--------	--

Description

Writes an FCS file for each sample from a dbFrame.

Usage

```
outFCS(x, y, out_path = tempdir(), ...)

## S4 method for signature 'dbFrame,flowFrame'
outFCS(x, y, out_path = tempdir(),
       out_nms = NULL, verbose = TRUE)
```

Arguments

x	a dbFrame .
y	a flowFrame containing the original measurement and meta data.
out_path	character string. Specifies in which location output files are to be generated.
...	optional arguments.
out_nms	an optional character string. Either the name of a 2 column CSV table with sample IDs and desired output file names, or a vector of length nrow(bc_key(x)) ordered as the samples in the barcoding scheme. If NULL (default), sample IDs will be used as file names.
verbose	if TRUE (default), a warning is given about populations for which no FCS files have been generated.

Details

Creates a separate FCS file for each barcode population. If out_nms is NULL (the default), files will be named after the barcode population's ID in the bc_key slot of the input [dbFrame](#); unassigned events will be written to "unassigned.fcs", and no output is generated for populations with less than 10 event assignments.

Value

a character of the output path.

Author(s)

Helena Lucia Crowell <helena.crowell@uzh.ch>

Examples

```
data(sample_ff, sample_key)
re <- assignPrelim(x = sample_ff, y = sample_key)
re <- estCutoffs(x = re)
re <- applyCutoffs(x = re)
outFCS(x = re, y = sample_ff)
```

outFrames	<i>Population-wise flowFrames from a dbFrame</i>
-----------	--

Description

Returns a `flowSet` or list of `flowFrames` from a `dbFrame`. Each `flowFrame` will contain the subset of events that have been assigned to the same ID.

Usage

```
outFrames(x, ...)  
  
## S4 method for signature 'dbFrame'  
outFrames(x, return = "flowSet",  
          which = "assigned")
```

Arguments

x	a <code>dbFrame</code> .
...	optional arguments.
return	"flowSet" or "list". Specifies the output type.
which	Specifies which barcode(s) to include. "assigned" (if the population of unassigned events should be excluded), "all" (if the latter should be included), or a numeric or character specifying a subset of populations. Valid values are IDs that occur as row names in the <code>bc_key</code> of the supplied <code>dbFrame</code> . Defaults to "assigned".

Details

Creates a separate `flowFrame` for each barcode population and, if desired, the population of unassigned events.

Value

a `flowSet` or list of `flowFrames`.

Author(s)

Helena Lucia Crowell <helena.crowell@uzh.ch>

Examples

```
data(sample_ff, sample_key)  
re <- assignPrelim(x = sample_ff, y = sample_key)  
re <- estCutoffs(x = re)  
re <- applyCutoffs(x = re)  
outFrames(x = re, return = "list", which = c("B1", "D4"))
```

plotAbundances *Population frequencies across samples & clusters*

Description

Plots the relative population abundances of the specified clustering.

Usage

```
plotAbundances(x, ...)

## S4 method for signature 'daFrame'
plotAbundances(x, k = "meta20", by = c("sample_id",
  "cluster_id"), group = NULL, shape = NULL)
```

Arguments

x	a daFrame .
...	optional arguments.
k	character string. Specifies which clustering to use.
by	a character string specifying whether to plot frequencies by samples or clusters.
group	a character string. Should corresponds to a column name of <code>rowData(x)</code> other than "sample_id" and "cluster_id". The default NULL will use the first factor available.
shape	a character string. Should correspond to a column name of <code>rowData(x)</code> other than "sample_id" and "cluster_id". The default NULL will use the first factor available.

Value

a ggplot object.

Author(s)

Helena Lucia Crowell <helena.crowell@uzh.ch>

References

Nowicka M, Krieg C, Weber LM et al. CyTOF workflow: Differential discovery in high-throughput high-dimensional cytometry datasets. *F1000Research* 2017, 6:748 (doi: 10.12688/f1000research.11622.1)

Examples

```
data(PBMC_fs, PBMC_panel, PBMC_md)
re <- daFrame(PBMC_fs, PBMC_panel, PBMC_md)
# run clustering
re <- cluster(re)
# plot relative population abundances
plotAbundances(re, k="meta12") # ...by sample
plotAbundances(re, k="meta8", by="cluster_id") # ...by cluster
```

plotClusterExprs *Plot expression distributions by cluster*

Description

Plots smoothed densities of arcsinh-transformed marker intensities by cluster.

Usage

```
plotClusterExprs(x, ...)  
  
## S4 method for signature 'daFrame'  
plotClusterExprs(x, k = "meta20", markers = NULL)
```

Arguments

x	a daFrame .
...	optional arguments.
k	character string. Specifies the clustering to use.
markers	character string specifying which markers to include. Defaults to NULL (= all markers). Alternatively, if the <code>colData(x)\$marker_class</code> column is specified, can be one of "type", "state", or "none".

Value

a [ggplot](#) object.

Author(s)

Helena Lucia Crowell <helena.crowell@uzh.ch>

References

Nowicka M, Krieg C, Weber LM et al. CyTOF workflow: Differential discovery in high-throughput high-dimensional cytometry datasets. *F1000Research* 2017, 6:748 (doi: 10.12688/f1000research.11622.1)

Examples

```
data(PBMC_fs, PBMC_panel, PBMC_md)  
re <- daFrame(PBMC_fs, PBMC_panel, PBMC_md)  
re <- cluster(re)  
plotClusterExprs(re)
```

plotClusterHeatmap *Plot cluster heatmap*

Description

Plots heatmaps summarizing a clustering and/or metaclustering of interest.

Usage

```
plotClusterHeatmap(x, ...)

## S4 method for signature 'daFrame'
plotClusterHeatmap(x, hm2 = NULL, k = "meta20",
  m = NULL, fun = c("median", "mean"), cluster_anno = TRUE,
  split_by = NULL, scale = TRUE, draw_dend = TRUE,
  draw_freqs = FALSE, palette = rev(brewer.pal(11, "RdYlBu")))
```

Arguments

x	a daFrame .
...	optional arguments.
hm2	character string. Specifies the right-hand side heatmap. One of: <ul style="list-style-type: none"> "abundances": cluster frequencies across samples "state_markers": median cell state marker expressions across clusters (analogous to the left-hand side heatmap) a character string/vector corresponding to one/multiple marker(s): median marker expressions across samples and clusters
k	character string. Specifies the clustering across which median marker expressions should be computed.
m	character string. Specifies the metaclustering to be shown. (This is for display only and will not effect any computations!)
fun	character string specifying the function to use as summary statistic.
cluster_anno	logical. Specifies if clusters should be annotated.
split_by	character string. Must corresponds to a column name of <code>rowData(x)</code> . If specified, the data will be subset according to this variable, and multiple heatmaps will be drawn.
scale	logical. Specifies whether scaled values should be plotted. (see below for details)
draw_dend	logical. Specifies if the row dendrogram should be drawn.
draw_freqs	logical. Specifies whether to display cell counts and proportions.
palette	character vector of colors to interpolate.

Details

Scaled values corresponds to cofactor arcsinh-transformed expression values scaled between 0 and 1 using 1 boundaries. Hierarchical clustering is performed on the unscaled data.

In its 1st panel, `plotClusterHeatmap` will display median (scaled, arcsinh-transformed) cell-type marker expressions (across all samples). Depending on argument `hm2`, the 2nd panel will contain one of:

- relative cluster abundances by sample
- median (scaled, arcsinh-transformed) cell-state marker expressions (across all samples)
- median (scaled, arcsinh-transformed) cell-state marker expressions by sample

Value

a `HeatmapList`-class object.

Author(s)

Helena Lucia Crowell <helena.crowell@uzh.ch>

References

Nowicka M, Krieg C, Weber LM et al. CyTOF workflow: Differential discovery in high-throughput high-dimensional cytometry datasets. *F1000Research* 2017, 6:748 (doi: 10.12688/f1000research.11622.1)

Examples

```
data(PBMC_fs, PBMC_panel, PBMC_md)
re <- daFrame(PBMC_fs, PBMC_panel, PBMC_md)

# run clustering
re <- cluster(re)

plotClusterHeatmap(re, hm2="abundances")
plotClusterHeatmap(re, hm2="abundances", draw_freqs=TRUE)
plotClusterHeatmap(re, hm2="state_markers", k="meta16", split_by='condition')
plotClusterHeatmap(re, hm2="pS6", k="meta12", m="meta8")
plotClusterHeatmap(re, hm2="abundances", scale=FALSE, draw_freqs=TRUE)
```

Description

Plots the tSNE and PCA representing the SOM codes as inferred by **FlowSOM**. Sizes are scaled to the total number of events assigned to each cluster, and points are color according to their cluster ID upon **ConsensusClusterPlus** metaclustering into k clusters.

Usage

```
plotCodes(x, ...)

## S4 method for signature 'daFrame'
plotCodes(x, k = "meta20", out_path = NULL,
          verbose = TRUE)
```

Arguments

x a [daFrame](#).

... optional arguments.

k character string. Specifies the clustering to use for color coding.

out_path character string. If specified, output will be generated in this location.

verbose logical. Specifies whether information on progress should be reported.

Value

a ggplot object.

Author(s)

Helena Lucia Crowell <helena.crowell@uzh.ch>

References

Nowicka M, Krieg C, Weber LM et al. CyTOF workflow: Differential discovery in high-throughput high-dimensional cytometry datasets. *F1000Research* 2017, 6:748 (doi: 10.12688/f1000research.11622.1)

Examples

```
data(PBMC_fs, PBMC_panel, PBMC_md)
re <- daFrame(PBMC_fs, PBMC_panel, PBMC_md)

# run clustering
re <- cluster(re)
plotCodes(re)
```

plotCounts

Plot cell counts

Description

Barplot of the number of cells measured for each sample.

Usage

```
plotCounts(x, ...)

## S4 method for signature 'daFrame'
plotCounts(x, color_by = "condition")
```

Arguments

x a `daFrame`.
 ... optional arguments.
 color_by character string. Must appear as a column name of `rowData(x)`. Specifies the color coding.

Value

a `ggplot` object.

Author(s)

Helena Lucia Crowell <helena.crowell@uzh.ch>

References

Nowicka M, Krieg C, Weber LM et al. CyTOF workflow: Differential discovery in high-throughput high-dimensional cytometry datasets. *F1000Research* 2017, 6:748 (doi: 10.12688/f1000research.11622.1)

Examples

```
data(PBMC_fs, PBMC_panel, PBMC_md)
re <- daFrame(PBMC_fs, PBMC_panel, PBMC_md)
plotCounts(re)
```

plotDiffHeatmap	<i>Plot differential heatmap</i>
-----------------	----------------------------------

Description

Heatmaps summarizing differential abundance & differential state testing results.

Usage

```
plotDiffHeatmap(x, y, ...)

## S4 method for signature 'matrix,SummarizedExperiment'
plotDiffHeatmap(x, y, top_n = 20,
  all = FALSE, order = TRUE, th = 0.1, hm1 = TRUE,
  normalize = TRUE, row_anno = TRUE, ...)

## S4 method for signature 'daFrame,SummarizedExperiment'
plotDiffHeatmap(x, y,
  top_n = 20, all = FALSE, order = TRUE, th = 0.1, hm1 = TRUE,
  normalize = TRUE, row_anno = TRUE, ...)

## S4 method for signature 'SummarizedExperiment,SummarizedExperiment'
plotDiffHeatmap(x, y,
  top_n = 20, all = FALSE, order = TRUE, th = 0.1, hm1 = TRUE,
  normalize = TRUE, row_anno = TRUE, ...)
```

```
## S4 method for signature 'ANY,list'
plotDiffHeatmap(x, y, top_n = 20, all = FALSE,
  order = TRUE, th = 0.1, hm1 = TRUE, normalize = TRUE,
  row_anno = TRUE, ...)
```

Arguments

x	a daFrame or <code>SummarizedExperiment</code> .
y	a <code>SummarizedExperiment</code> containing differential testing results as returned by one of testDA_edgeR , testDA_voom , testDA_GLMM , testDS_limma , or testDS_LMM . Alternatively, a list as returned by diffcyt .
...	optional arguments.
top_n	numeric. Number of top clusters (if type = "DA") or cluster-marker combinations (if type = "DS") to display.
all	logical. Specifies whether all clusters or cluster-marker combinations should be displayed. If TRUE, top_n will be ignored.
order	logical. Should results be ordered by significance?
th	numeric. Threshold on adjusted p-values below which clusters (DA) or cluster-marker combinations (DS) should be considered significant.
hm1	logical. Specifies whether the left-hand side heatmap should be plotted.
normalize	logical. Specifies whether Z-score normalized values should be plotted in the right-hand side heatmap. If y contains DA analysis results, relative population abundances will be arcsine-square-root scaled prior to normalization.
row_anno	logical. Should a row annotation indicating whether cluster (DA) or cluster-marker combinations (DS) are significant, as well as adjusted p-values be included?

Details

For DA tests, `plotDiffHeatmap` will display

- median (arcsinh-transformed) cell-type marker expressions (across all samples)
- cluster abundances by samples
- row annotations indicating if detected cluster are significant (i.e. adj. p-value \geq th)

For DS tests, `plotDiffHeatmap` will display

- median (arcsinh-transformed) cell-type marker expressions (across all samples)
- median (arcsinh-transformed) cell-state marker expressions by sample
- row annotations indicating if detected cluster-marker combinations are significant (i.e. adj. p-value \geq th)

Value

a [HeatmapList-class](#) object.

Author(s)

Lukas M Weber and Helena Lucia Crowell <helena.crowell@uzh.ch>

Examples

```

# construct daFrame
data(PBMC_fs, PBMC_panel, PBMC_md)
re <- daFrame(PBMC_fs, PBMC_panel, PBMC_md)

# run clustering
re <- cluster(re)

## differential analysis
library(diffcyt)

# create design & contrast matrix
design <- createDesignMatrix(PBMC_md, cols_design=3:4)
contrast <- createContrast(c(0, 1, 0, 0, 0))

# test for
# - differential abundance (DA) of clusters
# - differential states (DS) within clusters
da <- diffcyt(re, design = design, contrast = contrast,
  analysis_type = "DA", method_DA = "diffcyt-DA-edgeR",
  clustering_to_use = "meta20")
ds <- diffcyt(re, design = design, contrast = contrast,
  analysis_type = "DS", method_DS = "diffcyt-DS-limma",
  clustering_to_use = "meta20")

# display test results for
# - top DA clusters
# - top DS cluster-marker combinations
plotDiffHeatmap(re, da)
plotDiffHeatmap(re, ds)

```

plotDR

Plot dim. reduction from a daFrame

Description

Plot cell-level reduced dimensions stored within a daFrame object.

Usage

```

plotDR(x, ...)

## S4 method for signature 'daFrame'
plotDR(x, dr = c("TSNE", "PCA", "MDS", "UMAP",
  "DiffusionMap"), color_by = "meta20", facet = NULL)

```

Arguments

x	a daFrame .
...	optional arguments.
dr	character string specifying the dimensionality reduction method.

color_by character string specifying a clustering, marker, or rowData column name to color by.

facet a character string specifying a rowData column to facet by.

Value

a ggplot object.

Author(s)

Helena L. Crowell <helena.crowell@uzh.ch>

Examples

```
data(PBMC_fs, PBMC_panel, PBMC_md)
daf <- daFrame(PBMC_fs, PBMC_panel, PBMC_md)

daf <- runDR(daf, "PCA")
plotDR(daf, "PCA", color_by = "condition")

daf <- cluster(daf)
plotDR(daf, "PCA", color_by = "meta5")
```

plotEvents

Event plot

Description

Plots normalized barcode intensities for a given barcode.

Usage

```
plotEvents(x, ...)

## S4 method for signature 'dbFrame'
plotEvents(x, which = "all", n_events = 100,
           out_path = NULL, name_ext = NULL)
```

Arguments

x a [dbFrame](#).

... optional arguments.

which "all", numeric or character. Specifies which barcode(s) to plot. Valid values are IDs that occur as row names in the bc_key of the supplied [dbFrame](#), or 0 for unassigned events.

n_events numeric. Specifies number of events to plot. Defaults to 100.

out_path character string. If specified, outputs will be generated here.

name_ext character string. If specified, will be appended to the file name.

Value

Plots intensities normalized by population for each barcode specified by which: Each event corresponds to the intensities plotted on a vertical line at a given point along the x-axis. Events are scaled to the 95% quantile of the population it has been assigned to. Barcodes with less than 50 event assignments will be skipped; it is strongly recommended to remove such populations or reconsider their separation cutoffs.

Author(s)

Helena Lucia Crowell <helena.crowell@uzh.ch>

References

Zunder, E.R. et al. (2015). Palladium-based mass tag cell barcoding with a doublet-filtering scheme and single-cell deconvolution algorithm. *Nature Protocols* **10**, 316-333.

Examples

```
data(sample_ff, sample_key)

# view preliminary assignments
re <- assignPrelim(x = sample_ff, y = sample_key)
plotEvents(x = re, which = "D1", n_events = 1000)

# apply deconvolution parameters
re <- estCutoffs(re)
re <- applyCutoffs(x = re)
plotEvents(x = re, which = "D1", n_events = 500)
```

plotExprHeatmap	<i>Plot expression heatmap</i>
-----------------	--------------------------------

Description

Plots median marker expressions across samples computed on arcsinh-transformed intensities.

Usage

```
plotExprHeatmap(x, ...)

## S4 method for signature 'daFrame'
plotExprHeatmap(x, anno = TRUE, color_by = NULL,
  palette = brewer.pal(n = 8, name = "YlGnBu"), scale = TRUE,
  draw_freqs = FALSE, clustering_distance = "euclidean",
  clustering_linkage = "average")
```

Arguments

x	a daFrame .
...	optional arguments.
anno	logical. Specifies whether to display values inside each bin.
color_by	character string. Specifies the row annotation.
palette	character vector of colors to interpolate.
scale	logical. Specifies whether scaled values should be displayed. (see below for details)
draw_freqs	logical. Specifies whether to display cell counts and proportions.
clustering_distance	a character string that specifies the metric to use in dist() for clustering.
clustering_linkage	a character string that specifies the linkage to use in hclust() for clustering.

Details

Scaled values corresponds to cofactor arcsinh-transformed expression values scaled between 0 and 1 using 1 boundaries. Hierarchical clustering is performed on the unscaled data.

Value

a [HeatmapList-class](#) object.

Author(s)

Helena Lucia Crowell <helena.crowell@uzh.ch>

References

Nowicka M, Krieg C, Weber LM et al. CyTOF workflow: Differential discovery in high-throughput high-dimensional cytometry datasets. *F1000Research* 2017, 6:748 (doi: 10.12688/f1000research.11622.1)

Examples

```
data(PBMC_fs, PBMC_panel, PBMC_md)
daf <- daFrame(PBMC_fs, PBMC_panel, PBMC_md)
plotExprHeatmap(daf[, 1:5], color_by="condition", draw_freqs=TRUE)
```

plotExprs

Plot expressions

Description

Plots the smoothed densities of arcsinh-transformed marker intensities.

Usage

```
plotExprs(x, ...)  
  
## S4 method for signature 'daFrame'  
plotExprs(x, color_by = "condition")
```

Arguments

x	a daFrame .
...	optional arguments.
color_by	character string. Has to appear as a column name of rowData(x). Specifies the color coding.

Value

a [ggplot](#) object.

Author(s)

Helena Lucia Crowell <helena.crowell@uzh.ch>

References

Nowicka M, Krieg C, Weber LM et al. CyTOF workflow: Differential discovery in high-throughput high-dimensional cytometry datasets. *F1000Research* 2017, 6:748 (doi: 10.12688/f1000research.11622.1)

Examples

```
data(PBMC_fs, PBMC_panel, PBMC_md)  
re <- daFrame(PBMC_fs, PBMC_panel, PBMC_md)  
plotExprs(re)
```

plotMahal

Biaxial plot

Description

Histogram of counts and plot of yields as a function of separation cutoffs.

Usage

```
plotMahal(x, ...)  
  
## S4 method for signature 'dbFrame'  
plotMahal(x, which, cofactor = 50, out_path = NULL,  
          name_ext = NULL)
```

Arguments

x	a dbFrame .
...	optional arguments.
which	character string. Specifies which barcode to plot.
cofactor	numeric. Cofactor used for asinh transformation.
out_path	character string. If specified, outputs will be generated here.
name_ext	character string. If specified, will be appended to file name.

Value

Plots all inter-barcode interactions for the population specified by argument which. Events are colored by their Mahalanobis distance.

Author(s)

Helena Lucia Crowell <helena.crowell@uzh.ch>

References

Zunder, E.R. et al. (2015). Palladium-based mass tag cell barcoding with a doublet-filtering scheme and single-cell deconvolution algorithm. *Nature Protocols* **10**, 316-333.

Examples

```
data(sample_ff, sample_key)
re <- assignPrelim(x = sample_ff, y = sample_key)
re <- estCutoffs(x = re)
re <- applyCutoffs(x = re)
plotMahal(x = re, which = "B3")
```

plotMDS

MDS plot

Description

Multi-dimensional scaling (MDS) plot on median marker expressions.

Usage

```
plotMDS(x, ...)

## S4 method for signature 'daFrame'
plotMDS(x, color_by = "condition")
```

Arguments

x	a daFrame .
...	optional arguments.
color_by	character string that appears as a column name of rowData(x). Specifies the color coding.

Value

a ggplot object.

Author(s)

Helena Lucia Crowell <helena.crowell@uzh.ch>

References

Nowicka M, Krieg C, Weber LM et al. CyTOF workflow: Differential discovery in high-throughput high-dimensional cytometry datasets. *F1000Research* 2017, 6:748 (doi: 10.12688/f1000research.11622.1)

Examples

```
data(PBMC_fs, PBMC_panel, PBMC_md)
re <- daFrame(PBMC_fs, PBMC_panel, PBMC_md)
plotMDS(re)
```

plotMedExprs

Plot median expressions

Description

Plots median marker expressions across samples computed on arcsinh-transformed intensities.

Usage

```
plotMedExprs(x, ...)
```

S4 method for signature 'daFrame'

```
plotMedExprs(x, k = "meta20", facet = c("antigen",
    "cluster_id"), group_by = "condition")
```

Arguments

x	a daFrame .
...	optional arguments.
k	character string. Specifies the clustering to use. If facet = "antigen", this argument will be ignored.
facet	"antigen" or "cluster_id". Note that the latter requires having run cluster first.
group_by	character string. Has to appear as a column name of rowData(x). Specifies sample grouping.

Value

a ggplot object.

Author(s)

Helena Lucia Crowell <helena.crowell@uzh.ch>

References

Nowicka M, Krieg C, Weber LM et al. CyTOF workflow: Differential discovery in high-throughput high-dimensional cytometry datasets. *F1000Research* 2017, 6:748 (doi: 10.12688/f1000research.11622.1)

Examples

```
data(PBMC_fs, PBMC_panel, PBMC_md)
re <- daFrame(PBMC_fs, PBMC_panel, PBMC_md)

# plot median expressions
plotMedExprs(re)

# run clustering
re <- cluster(re)

# plot median expressions across clusters
plotMedExprs(re, facet="cluster_id", k="meta8")
```

plotNRS

Plot non-redundancy scores

Description

Plots non-redundancy scores (NRS) by markers in decreasing order.

Usage

```
plotNRS(x, ...)

## S4 method for signature 'daFrame'
plotNRS(x, markers = NULL, color_by = "condition")
```

Arguments

x	a daFrame .
...	optional arguments.
markers	character string specifying which markers to include. Defaults to NULL (= all markers). Alternatively, if the <code>colData(x)\$marker_class</code> column is specified, can be one of "type", "state", or "none".
color_by	character string. Has to appear as a column name of <code>rowData(x)</code> . Specifies the color coding.

Value

a ggplot object.

Author(s)

Helena Lucia Crowell <helena.crowell@uzh.ch>

References

Nowicka M, Krieg C, Weber LM et al. CyTOF workflow: Differential discovery in high-throughput high-dimensional cytometry datasets. *F1000Research* 2017, 6:748 (doi: 10.12688/f1000research.11622.1)

Examples

```
data(PBMC_fs, PBMC_panel, PBMC_md)
re <- daFrame(PBMC_fs, PBMC_panel, PBMC_md)
plotNRS(re)
```

plotSpillmat	<i>Spillover matrix heat map</i>
--------------	----------------------------------

Description

Generates a heat map of the spillover matrix annotated with estimated spill percentages.

Usage

```
plotSpillmat(bc_ms, SM, ...)

## S4 method for signature 'numeric,matrix'
plotSpillmat(bc_ms, SM, out_path = NULL,
  name_ext = NULL, annotate = TRUE, plotly = TRUE,
  isotope_list = NULL)

## S4 method for signature 'ANY,data.frame'
plotSpillmat(bc_ms, SM, out_path = NULL,
  name_ext = NULL, annotate = TRUE, plotly = TRUE,
  isotope_list = NULL)

## S4 method for signature 'character,ANY'
plotSpillmat(bc_ms, SM, out_path = NULL,
  name_ext = NULL, annotate = TRUE, plotly = TRUE,
  isotope_list = NULL)
```

Arguments

bc_ms	a vector of numeric masses corresponding to barcode channels.
SM	spillover matrix returned from computeSpillmat.
...	optional arguments.
out_path	character string. If specified, outputs will be generated here.
name_ext	character string. If specified, will be appended to the plot's name.
annotate	logical. If TRUE (default), spill percentages are shown inside bins and rows are annotated with the total amount of spill received.

plotly logical. Should an interactive plot be rendered?
 isotope_list named list. Used to validate the input spillover matrix. Names should be metals;
 list elements numeric vectors of their isotopes. See [isotope_list](#) for the list of
 isotopes used by default.

Value

Plots estimated spill percentages as a heat map. Colours are ramped to the highest spillover value present

Author(s)

Helena Lucia Crowell <helena.crowell@uzh.ch>

Examples

```
# get single-stained control samples
data(ss_exp)

# specify mass channels stained for
bc_ms <- c(139, 141:156, 158:176)

re <- assignPrelim(x = ss_exp, y = bc_ms)
re <- estCutoffs(x = re)
re <- applyCutoffs(x = re)
spillMat <- computeSpillmat(x = re)
plotSpillmat(bc_ms = bc_ms, SM = spillMat)
```

plotYields

Yield plot

Description

Distribution of barcode separations and yields as a function of separation cutoffs.

Usage

```
plotYields(x, ...)
```

S4 method for signature 'dbFrame'

```
plotYields(x, which = 0, out_path = NULL,
           name_ext = NULL, plotly = TRUE)
```

Arguments

x a [dbFrame](#).
 ... optional arguments.
 which 0, numeric or character. Specifies which barcode(s) to plot. Valid values are IDs that occur as row names of `bc_key(x)`; 0 (the default) will generate a summary plot with all barcodes.
 out_path character string. If specified, outputs will be generated here.

Arguments

x	a daFrame .
...	additional parameters passed to the dim. reduction method (see runTSNE , runPCA , runMDS , runUMAP , and runDiffusionMap).
dr	character string specifying the dimensionality reduction method.
rows_to_use	numeric vector of row indices (cells) to use. If NULL, all cells will be used. If a single integer value N, (default 1000) a subset of N cells will be drawn from each sample.
cols_to_use	character vector in <code>colnames(x)</code> or numeric vector of column indices to use for computing reduced dimensions.
overwrite	logical. Whether to force overwriting any existing dimension reduction of type dr.

Value

a [daFrame](#) with an additional entry titled "dr" in the `reducedDims` slot of the input [daFrame](#).

Author(s)

Helena L. Crowell <helena.crowell@uzh.ch>

Examples

```
data(PBMC_fs, PBMC_panel, PBMC_md)
daf <- daFrame(PBMC_fs, PBMC_panel, PBMC_md)
daf <- cluster(daf)

# PCA on all cells
daf <- runDR(daf, "PCA")

# UMAP on 1000 random cells
daf <- runDR(daf, "UMAP", rows_to_use = sample(nrow(daf), 1e3))

reducedDims(daf)
head(reducedDim(daf, "UMAP"))

# PCA on 200 cells per sample
set.seed(1)
daf <- runDR(daf, "PCA", rows_to_use = 200, overwrite = TRUE)

# re-using PCA for t-SNE will fail when using different cells
## Not run:
daf <- runDR(daf, "TSNE", rows_to_use = 1:500, use_dimred = "PCA")
## End(Not run)

# use same seed to assure the same subset of cells is sampled
set.seed(1)
daf <- runDR(daf, "TSNE", rows_to_use = 200, use_dimred = "PCA")

# number of rows used for each DR:
vapply(reducedDims(daf), nrow, numeric(1))

# running on subset can be done 2-ways
daf2 <- runDR(daf, "MDS", 1:100)
```

```
daf3 <- runDR(daf[1:100, ], "MDS")

# option 1 keeps object-dimensions
identical(dim(daf2), dim(daf))

# option 2 keeps only specified rows
all.equal(dim(daf3), c(100, ncol(daf)))

# reduced dimension are identical
identical(reducedDim(daf2), reducedDim(daf2))

# run t-SNE on B-cell clusters only
data(merging_table)
daf <- mergeClusters(daf, "meta20", merging_table, "merging")
cells_use <- grep("B-cells", cluster_ids(daf, "merging"))
daf <- runDR(daf, "TSNE", cells_use, overwrite = TRUE)
plotDR(daf, "TSNE", "merging")
```

Index

- [, daFrame, ANY, ANY, ANY-method (ei), 18
- adaptSpillmat, 3
- adaptSpillmat, matrix, vector-method (adaptSpillmat), 3
- applyCutoffs, 4, 15, 17
- applyCutoffs, dbFrame-method (applyCutoffs), 4
- assignPrelim, 5, 15, 17
- assignPrelim, character, data.frame-method (assignPrelim), 5
- assignPrelim, character, vector-method (assignPrelim), 5
- assignPrelim, flowFrame, data.frame-method (assignPrelim), 5
- assignPrelim, flowFrame, vector-method (assignPrelim), 5
- bc_ids (dbFrame-methods), 16
- bc_ids, dbFrame-method (dbFrame-methods), 16
- bc_key (dbFrame-methods), 16
- bc_key, dbFrame-method (dbFrame-methods), 16
- cluster, 6, 43
- cluster, daFrame-method (cluster), 6
- cluster_codes (ei), 18
- cluster_codes, daFrame-method (ei), 18
- cluster_ids (ei), 18
- cluster_ids, daFrame-method (ei), 18
- compCytof, 3, 7
- compCytof, ANY, data.frame-method (compCytof), 7
- compCytof, character, matrix-method (compCytof), 7
- compCytof, flowFrame, matrix-method (compCytof), 7
- compCytof, flowSet, ANY-method (compCytof), 7
- computeSpillmat, 9
- computeSpillmat, dbFrame-method (computeSpillmat), 9
- concatFCS, 10
- concatFCS, character-method (concatFCS), 10
- concatFCS, flowSet-method (concatFCS), 10
- concatFCS, list-method (concatFCS), 10
- ConsensusClusterPlus, 19
- counts (dbFrame-methods), 16
- counts, dbFrame-method (dbFrame-methods), 16
- daFrame, 6, 18, 19, 22, 23, 25, 30–32, 34–37, 40–44, 48
- daFrame (daFrame-class), 12
- daFrame, character, data.frame, data.frame-method (daFrame-class), 12
- daFrame, daFrame, ANY, matrix-method (daFrame-class), 12
- daFrame, daFrame, matrix, ANY-method (daFrame-class), 12
- daFrame, flowSet, data.frame, data.frame-method (daFrame-class), 12
- daFrame-class, 12
- daFrame-methods (ei), 18
- data, 13
- dbFrame, 4, 6, 9, 16, 17, 20, 21, 28, 29, 38, 42, 46
- dbFrame (dbFrame-class), 15
- dbFrame-class, 15
- dbFrame-methods, 16
- deltas (dbFrame-methods), 16
- deltas, dbFrame-method (dbFrame-methods), 16
- diffcyt, 36
- dist, 40
- ei, 18
- ei, daFrame-method (ei), 18
- estCutoffs, 15, 20
- estCutoffs, dbFrame-method (estCutoffs), 20
- exprs (ei), 18
- exprs, daFrame-method (ei), 18
- exprs, dbFrame-method (dbFrame-methods), 16
- extractClusters, 22

- extractClusters, daFrame, character-method
(extractClusters), 22
- filter, 23
- filter, daFrame, character-method
(filter), 23
- filter, daFrame, missing-method (filter),
23
- flowFrame, 5, 8, 13, 27–29
- flowSet, 13, 14, 27, 29
- FlowSOM, 19
- ggplot, 7, 31, 35, 41
- guessPanel, 24
- guessPanel, flowFrame-method
(guessPanel), 24
- guessPanel, flowSet-method (guessPanel),
24
- hclust, 40
- isotope_list, 3, 8, 10, 46
- isotope_list (data), 13
- launchGUI, 25
- marker_classes (ei), 18
- marker_classes, daFrame-method (ei), 18
- mergeClusters, 12, 19, 25
- mergeClusters, daFrame, character, data.frame, character-method
(mergeClusters), 25
- merging_table (data), 13
- mhl_cutoff (dbFrame-methods), 16
- mhl_cutoff, dbFrame-method
(dbFrame-methods), 16
- mhl_cutoff<- (dbFrame-methods), 16
- mhl_cutoff<-, dbFrame, ANY-method
(dbFrame-methods), 16
- mhl_cutoff<-, dbFrame, numeric-method
(dbFrame-methods), 16
- mhl_dists (dbFrame-methods), 16
- mhl_dists, dbFrame-method
(dbFrame-methods), 16
- mp_cells (data), 13
- n_cells (ei), 18
- n_cells, daFrame-method (ei), 18
- normCytof, 26
- normCytof, character-method (normCytof),
26
- normCytof, flowFrame-method (normCytof),
26
- normed_bcs (dbFrame-methods), 16
- normed_bcs, dbFrame-method
(dbFrame-methods), 16
- outFCS, 28
- outFCS, dbFrame, flowFrame-method
(outFCS), 28
- outFrames, 29
- outFrames, dbFrame-method (outFrames), 29
- PBMC_fs (data), 13
- PBMC_md (data), 13
- PBMC_panel (data), 13
- plotAbundances, 26, 30
- plotAbundances, daFrame-method
(plotAbundances), 30
- plotClusterExprs, 31
- plotClusterExprs, daFrame-method
(plotClusterExprs), 31
- plotClusterHeatmap, 26, 32
- plotClusterHeatmap, daFrame-method
(plotClusterHeatmap), 32
- plotCodes, 33
- plotCodes, daFrame-method (plotCodes), 33
- plotCounts, 34
- plotCounts, daFrame-method (plotCounts),
34
- plotDiffHeatmap, 35
- plotDiffHeatmap, ANY, list-method
(plotDiffHeatmap), 35
- plotDiffHeatmap, daFrame, SummarizedExperiment-method
(plotDiffHeatmap), 35
- plotDiffHeatmap, matrix, SummarizedExperiment-method
(plotDiffHeatmap), 35
- plotDiffHeatmap, SummarizedExperiment, SummarizedExperiment-method
(plotDiffHeatmap), 35
- plotDR, 37
- plotDR, daFrame-method (plotDR), 37
- plotEvents, 15, 38
- plotEvents, dbFrame-method (plotEvents),
38
- plotExprHeatmap, 39
- plotExprHeatmap, daFrame-method
(plotExprHeatmap), 39
- plotExprs, 40
- plotExprs, daFrame-method (plotExprs), 40
- plotMahal, 15, 41
- plotMahal, dbFrame-method (plotMahal), 41
- plotMDS, 42
- plotMDS, daFrame-method (plotMDS), 42
- plotMedExprs, 43
- plotMedExprs, daFrame-method
(plotMedExprs), 43
- plotNRS, 44

plotNRS, daFrame-method (plotNRS), 44
plotSpillmat, 45
plotSpillmat, ANY, data.frame-method
 (plotSpillmat), 45
plotSpillmat, character, ANY-method
 (plotSpillmat), 45
plotSpillmat, numeric, matrix-method
 (plotSpillmat), 45
plotYields, 15, 46
plotYields, dbFrame-method (plotYields),
 46

raw_data (data), 13
reducedDim, daFrame-method (ei), 18
reducedDimNames, daFrame-method (ei), 18
reducedDimNames<-, daFrame, character-method
 (ei), 18
reducedDims, daFrame-method (ei), 18
runDiffusionMap, 48
runDR, 47
runDR, daFrame-method (runDR), 47
runMDS, 48
runPCA, 48
runTSNE, 48
runUMAP, 48

sample_ff (data), 13
sample_ids (ei), 18
sample_ids, daFrame-method (ei), 18
sample_key (data), 13
sep_cutoffs (dbFrame-methods), 16
sep_cutoffs, dbFrame-method
 (dbFrame-methods), 16
sep_cutoffs<- (dbFrame-methods), 16
sep_cutoffs<-, dbFrame, ANY-method
 (dbFrame-methods), 16
sep_cutoffs<-, dbFrame, numeric-method
 (dbFrame-methods), 16
ss_exp (data), 13
state_markers (ei), 18
state_markers, daFrame-method (ei), 18

testDA_edgeR, 36
testDA_GLMM, 36
testDA_voom, 36
testDS_limma, 36
testDS_LMM, 36
type_markers (ei), 18
type_markers, daFrame-method (ei), 18

yields (dbFrame-methods), 16
yields, dbFrame-method
 (dbFrame-methods), 16