Package 'CATALYST'

November 11, 2026

Type Package

Title Cytometry dATa anALYSis Tools

Version 1.36.0

Depends R (>= 4.6), SingleCellExperiment

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

Description CATALYST provides tools for preprocessing of and differential discovery in cytometry data such as FACS, CyTOF, and IMC. Preprocessing includes i) normalization using bead standards, ii) single-cell deconvolution, and iii) bead-based compensation.
For differential discovery, the package provides a number of convenient functions for data processing (e.g., clustering, dimension reduction), as well as a suite of visualizations for exploratory data analysis and exploration of results from differential abundance (DA) and state (DS) analysis in order to identify differences in composition and expression profiles at the subpopulation-level, respectively.

Imports circlize, ComplexHeatmap, ConsensusClusterPlus, cowplot, dplyr, drc, flowCore, FlowSOM, ggplot2, ggrepel, ggridges, graphics, grDevices, grid, gridExtra, Matrix, matrixStats, methods, nnls, RColorBrewer, reshape2, Rtsne, SummarizedExperiment, S4Vectors, scales, scater, stats

Suggests BiocStyle, diffcyt, flowWorkspace, ggcyto, knitr, openCyto, rmarkdown, testthat, uwot

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

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

VignetteBuilder knitr

RoxygenNote 7.3.3

License GPL (>=2)

Encoding UTF-8

LazyData true

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

git_branch RELEASE_3_23

git_last_commit 093e4f5

git_last_commit_date 2026-04-28

Repository Bioconductor 3.23

Date/Publication 2026-11-11

Author Helena L. Crowell [aut, cre] (ORCID:
<https://orcid.org/0000-0002-4801-1767>),
Vito R.T. Zanotelli [aut] (ORCID:
<https://orcid.org/0000-0001-7268-311X>),
Stéphane Chevrier [aut, dtc] (ORCID:
<https://orcid.org/0000-0002-9216-7910>),
Mark D. Robinson [aut, fnd] (ORCID:
<https://orcid.org/0000-0002-3048-5518>),
Bernd Bodenmiller [fnd] (ORCID:
<https://orcid.org/0000-0002-6325-7861>)

Maintainer Helena L. Crowell <helena@crowell.eu>

Contents

adaptSpillmat . . . 3
applyCutoffs . . . 4
assignPrelim . . . 5
clrDR . . . 6
cluster . . . 9
compCytof . . . 11
computeSpillmat . . . 13
data . . . 15
estCutoffs . . . 16
extractClusters . . . 18
filterSCE . . . 19
guessPanel . . . 20
mergeClusters . . . 21
normCytof . . . 22
pbMDS . . . 24
plotAbundances . . . 26
plotClusterExprs . . . 28
plotCodes . . . 29
plotCounts . . . 30
plotDiffHeatmap . . . 31
plotDR . . . 33
plotEvents . . . 35
plotExprHeatmap . . . 37
plotExprs . . . 40

plotFreqHeatmap 41
plotMahal 43
plotMultiHeatmap 44
plotNRS 46
plotPbExprs 47
plotScatter 49
plotSpillmat 51
plotYields 52
prepData 53
runDR 56
SCE-accessors 57
sce2fcs 59

Index **61**

`adaptSpillmat` *Adapt spillover matrix*

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(
  x,
  out_chs,
  isotope_list = CATALYST::isotope_list,
  verbose = TRUE
)
```

Arguments

`x`	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. Ir191, Ir191Di or Ir191(Di), will be interpreted as masses with associated metals.
`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.
`verbose`	logical. Should warnings about possibly inaccurate spillover estimates be printed to the console?

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 L Crowell <helena.crowell@uzh.ch> & Vito RT Zanotelli

Examples

```
# estimate spillover matrix from
# single-stained control samples
data(ss_exp)
sce <- prepData(ss_exp)
bc_ms <- c(139, 141:156, 158:176)
sce <- assignPrelim(sce, bc_ms, verbose = FALSE)
sce <- applyCutoffs(estCutoffs(sce))
sce <- computeSpillmat(sce)

library(SingleCellExperiment)
sm1 <- metadata(sce)$spillover_matrix
sm2 <- adaptSpillmat(sm1, rownames(sce), verbose = FALSE)
all(dim(sm2) == ncol(sm1))
```

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

Description

Applies separation and mahalanobies distance cutoffs.

Usage

```
applyCutoffs(x, assay = "exprs", mhl_cutoff = 30, sep_cutoffs = NULL)
```

Arguments

`x`	a `SingleCellExperiment`.
`assay`	character string specifying which assay data to use. Should be one of `assayNames(x)` and correspond to expression-like not count data.
`mhl_cutoff`	numeric mahalanobis distance threshold above which events should be unassigned; ignored if `metadata(x)$mhl_cutoff` exists.
`sep_cutoffs`	non-negative numeric of length one or of same length as the number of rows in the `bc_key(x)`. Specifies the distance separation cutoffs between positive and negative barcode populations below which events should be unassigned. If `NULL` (default), `applyCutoffs` will try to access `metadata(x)$sep_cutoffs`.

Value

the input `SingleCellExperiment` `x` is returned with updated `colData` columns `"bc_id"` and `"mhl_dist"`, and an additional `int_metadata` slot `"mhl_cutoff"` containing the applied mahalanobies distance cutoff.

Author(s)

Helena L 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

```
library(SingleCellExperiment)

# construct SCE
data(sample_ff, sample_key)
sce <- prepData(sample_ff)

# assign preliminary barcode IDs
# & estimate separation cutoffs
sce <- assignPrelim(sce, sample_key)
sce <- estCutoffs(sce)

# use estimated population-specific
# vs. global separation cutoff(s)
sce1 <- applyCutoffs(sce)
sce2 <- applyCutoffs(sce, sep_cutoffs = 0.35)

# compare yields after applying cutoff(s)
c(global = mean(sce1$bc_id != 0),
specific = mean(sce2$bc_id != 0))
```

---

assignPrelim *Single-cell debarcoding (1)*

Description

Assigns a preliminary barcode ID to each event.

Usage

```
assignPrelim(x, bc_key, assay = "exprs", verbose = TRUE)
```

Arguments

x	a SingleCellExperiment.
bc_key	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.
assay	character string specifying which assay to use.
verbose	logical. Should extra information on progress be reported?

Value

a SingleCellExperiment structured as follows:

assays
- counts - raw counts
- exprs - arcsinh-transformed counts
- scaled - population-wise scaled expression using (95%)-quantiles as boundaries

colData
- bc_id - numeric vector of barcode assignments
- delta - separation between positive and negative barcode populations

metadata
- bc_key - the input debarcoding scheme

Author(s)

Helena L 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)
sce <- prepData(sample_ff)
sce <- assignPrelim(sce, sample_key)
table(sce$bc_id)
```

---

clrDR *DR plot on CLR of proportions*

Description

Computes centered log-ratios (CLR) on cluster/sample proportions across samples/clusters, and visualizes them in a lower-dimensional space, highlighting differences in composition between samples/clusters.

Usage

```
clrDR(
  x,
  dr = c("PCA", "MDS", "UMAP", "TSNE", "DiffusionMap"),
  by = c("sample_id", "cluster_id"),
  k = "meta20",
  dims = c(1, 2),
  base = 2,
  arrows = TRUE,
  point_col = switch(by, sample_id = "condition", "cluster_id"),
  arrow_col = switch(by, sample_id = "cluster_id", "condition"),
  arrow_len = 0.5,
  arrow_opa = 0.5,
  label_by = NULL,
  size_by = TRUE,
  point_pal = NULL,
  arrow_pal = NULL
)
```

Arguments

x	a SingleCellExperiment.
dr	character string specifying which dimension reduction to use.
by	character string specifying across which IDs to compute CLRs • by = "sample_id" compute CLRs across relative abundances of samples across clusters; each point in the embedded space represents a sample. • by = "cluster_id" compute CLRs across relative abundances of clusters across samples; each point in the embedded space represents a cluster.
k	character string specifying which clustering to use; valid values are names(cluster_codes(x)).
dims	two numeric scalars indicating which dimensions to plot.
base	integer scalar specifying the logarithm base to use.
arrows	logical specifying whether to include arrows for PC loadings.
point_col, arrow_col	character string specifying a non-numeric cell metadata column to color points and PC loading arrows by; valid values are names(colData(x)).
arrow_len	non-zero single numeric specifying the length of loading vectors relative to the largest xy-coordinate in the embedded space; NULL for no re-sizing (see details).
arrow_opa	single numeric in [0,1] specifying the opacity (alpha) of PC loading arrows when they are grouped; 0 will hide individual arrows.
label_by	character string specifying a non-numeric sample metadata variable to label points by; valid values are names(colData(x)).
size_by	logical specifying whether to scale point sizes by the number of cells in a given sample/cluster (for by = "sample/cluster_id").

`point_pal`, `arrow_pal`
: character string of colors to use for points and PC loading arrows. Arguments default to `.cluster_cols` for clusters (defined internally), and `brewer.pal`'s `"Set3"` for samples.

Details

The centered log-ratio (CLR) Let `k` be one of S samples, `k` one of K clusters, and `p(s,k)` be the proportion of cells from `s` in k. The centered log-ratio (CLR) is defined as

$$clr(sk) = logp(s,k) - \sum p(s,k)/K$$

and analogous for clusters replacing `s` by `k` and `K` by `S`. Thus, each sample/cluster gives a vector with length `K/S` and mean 0, and the CLRs computed across all instances can be represented as a matrix with dimensions `S` x `K` (or `K` x `S` for clusters) that we embed into a lower dimensional space.

Dimensionality reduction In principle, `clrDR` allows any dimension reduction to be applied on the CLRs. The default method (`dr = "PCA"`) will include the percentage of variance explained by each principal component (PC) in the axis labels.

Noteworthily, distances between points in the lower-dimensional space are meaningful only for linear DR methods (PCA and MDS), and results obtained from other methods should be interpreted with caution. Thus, the output plot's aspect ratio should be kept as is for PCA and MDS; non-linear DR methods can use `aspect.ratio = 1`, rendering a square plot.

Interpreting PC loadings For `dr = "PCA"`, PC loadings will be represented as arrows that may be interpreted as follows: 0° (180°) between vectors indicates a strong positive (negative) relation between them, while vectors that are orthogonal to each another (90°) are roughly independent.

When a vector points towards a given quadrant, the variability in proportions for the points within this quadrant are largely driven by the corresponding variable. Here, only the relative orientation of vectors to one another and to the PC axes is meaningful; however, the sign of loadings (i.e., whether an arrow points left or right) can be flipped when re-computing PCs.

When `arrow_len` is specified, PC loading vectors will be re-scaled to improve their visibility. Here, a value of 1 will stretch vectors such that the largest loading will touch on the outer most point. Importantly, while absolute arrow lengths are not interpretable, their relative length is.

Value

a `ggplot` object.

Author(s)

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

Examples

```
data(PBMC_fs, PBMC_panel, PBMC_md)
sce <- prepData(PBMC_fs, PBMC_panel, PBMC_md)
sce <- cluster(sce)
```

```
# CLR on sample proportions across clusters
# (1st vs. 3rd PCA; include sample labels)
clrDR(sce, by = "sample_id", k = "meta12",
  dims = c(1, 3), label_by = "sample_id")

# CLR on cluster proportions across samples
# (use custom colors for both points & loadings)
clrDR(sce, by = "cluster_id",
  point_pal = hcl.colors(10, "Spectral"),
  arrow_pal = c("royalblue", "orange"))
```

`cluster` `FlowSOM` *clustering &* `ConsensusClusterPlus` *metaclustering*

Description

`cluster` will first group cells into `xdim`x`ydim` clusters using **FlowSOM**, and subsequently perform metaclustering with **ConsensusClusterPlus** into 2 through `maxK` clusters.

Usage

```
cluster(
  x,
  features = "type",
  xdim = 10,
  ydim = 10,
  maxK = 20,
  verbose = TRUE,
  seed = 1
)
```

Arguments

`x`	a `SingleCellExperiment`.
`features`	a character vector specifying which features to use for clustering; valid values are `"type"`/`"state"` for `type/state_markers(x)` if `rowData(x)$marker_class` have been specified; a subset of `rownames(x)`; NULL to use all features.
`xdim`, `ydim`	numeric specifying the grid size of the self-orginizing map; passed to `BuildSOM`. The default 10x10 grid will yield 100 clusters.
`maxK`	numeric specifying the maximum number of clusters to evaluate in the metaclustering; passed to `ConsensusClusterPlus`. The default (`maxK = 20`) will yield 2 through 20 metaclusters.
`verbose`	logical. Should information on progress be reported?
`seed`	numeric. Sets the random seed for reproducible results 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 corresponds to the value of k where there is no longer a considerable increase in stability (pleateau onset).

Value

a `SingleCellEcperiment` with the following newly added data:

- `colData`
  - `cluster_id`: each cell's cluster ID as inferred by `FlowSOM`. One of 1, ..., `xdimxydim`.
- `rowData`
  - `marker_class`: added when previosly unspecified. `"type"` when an antigen has been used for clustering, otherwise `"state"`.
  - `used_for_clustering`: logical indicating whether an antigen has been used for clustering.
- `metadata`
  - `SOM_codes`: a table with dimensions K x (# cell type markers), where K = `xdim` x `ydim`. Contains the SOM codes.
  - `cluster_codes`: a table with dimensions K x (`maxK` + 1). Contains the cluster codes for all metaclustering.
  - `delta_area`: a `ggplot` object (see details).

Author(s)

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

References

Nowicka M, Krieg C, Crowell HL, 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

```
# construct SCE
data(PBMC_fs, PBMC_panel, PBMC_md)
sce <- prepData(PBMC_fs, PBMC_panel, PBMC_md)

# run clustering
(sce <- cluster(sce))

# view all available clustering
names(cluster_codes(sce))

# access specific clustering resolution
```

```
table(cluster_ids(sce, "meta8"))

# view delta area plot
delta_area(sce)
```

`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,
  sm = NULL,
  method = c("nnls", "flow"),
  assay = "counts",
  overwrite = TRUE,
  transform = TRUE,
  cofactor = NULL,
  isotope_list = CATALYST::isotope_list
)
```

Arguments

`x`	a `SingleCellExperiment` OR a character string specifying the location of FCS files that should be compensates.
`sm`	a spillover matrix.
`method`	`"flow"` or `"nnls"`.
`assay`	character string specifying which assay data to use; should be one of `assayNames(x)` and correspond to count-like data, as linearity assumptions underlying compensation won't hold otherwise.
`overwrite`	logical; should the specified `assay` slot (and `exprs`, when `transform = TRUE`) be overwritten with the compensated data? If `FALSE`, compensated counts (and expressions, if `transform = TRUE`) will be stored in assay(s) `compcounts/exprs`, respectively.
`transform`	logical; should normalized counts be arcsinh-transformed with the specified `cofactor`(s)?
`cofactor`	numeric cofactor(s) to use for optional arcsinh-transformation when `transform = TRUE`; single value or a vector with channels as names. If NULL, `compCytof` will try and access the cofactor(s) stored in `int_metadata(x)`, thus re-using the same transformation applied previously.

`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.

Author(s)

Helena L Crowell <helena.crowell@uzh.ch> & Vito RT Zanotelli

Examples

```
# deconvolute single-stained control samples
data(ss_exp)
sce <- prepData(ss_exp)
bc_ms <- c(139, 141:156, 158:176)
sce <- assignPrelim(sce, bc_ms)
sce <- applyCutoffs(estCutoffs(sce))

# estimate spillover matrix
sce <- computeSpillmat(sce)

# compensate & store compensated data in separate assays
sce <- compCytof(sce, overwrite = FALSE)
assayNames(sce)

# biscatter before vs. after compensation
chs <- c("Dy162Di", "Dy163Di")
m <- match(chs, channels(sce))
i <- rownames(sce)[m][1]
j <- rownames(sce)[m][2]

par(mfrow = c(1, 2))
for (a in c("exprs", "compexprs")) {
  es <- assay(sce, a)
```

```
  plot(es[i, ], es[j, ], cex = 0.2, pch = 19,
       main = a, xlab = i, ylab = j)
}
```

`computeSpillmat` *Compute spillover matrix*

Description

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

Usage

```
computeSpillmat(
  x,
  assay = "counts",
  interactions = c("default", "all"),
  method = c("default", "classic"),
  trim = 0.5,
  th = 1e-05
)
```

Arguments

`x`	a `SingleCellExperiment`.
`assay`	character string specifying which assay to use; should be one of `assayNames(x)` and correspond to count-like data, as linearity assumptions underlying spillover estimation won't hold otherwise.
`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).
`method`	`"default"` or `"classic"`. Specifies the function to be used for spillover estimation (see below for 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 impurites (see reference below and `isotope_list`).

`interaction="all"` will estimate spill for all n x 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 L 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

```
# construct SCE from single-stained control samples
data(ss_exp)
sce <- prepData(ss_exp)

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

# debarcode single-positive populations
sce <- assignPrelim(sce, bc_ms)
sce <- estCutoffs(sce)
sce <- applyCutoffs(sce)

# estimate & extract spillover matrix
sce <- computeSpillmat(sce)

library(SingleCellExperiment)
head(metadata(sce)$spillover_matrix)
```

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 L 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)
```

---

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)
```

Arguments

x	a SingleCellExperiment.

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:

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

and the cutoffs for both functions are defined as:

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

$$c_{log-logistic} = argmin_x\{\frac{|f'(x)|}{f(x)} > 0.1\}$$

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

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

Value

the input `SingleCellExperiment` is returned with an additional `metadata` slot `sep_cutoffs`.

Author(s)

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

References

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

Examples

```
library(SingleCellExperiment)

# construct SCE
data(sample_ff, sample_key)
sce <- prepData(sample_ff)

# assign preliminary barcode IDs
# & estimate separation cutoffs
sce <- assignPrelim(sce, sample_key)
sce <- estCutoffs(sce)
```

```
# access separation cutoff estimates
(seps <- metadata(sce)$sep_cutoffs)

# compute population yields
cs <- split(seq_len(ncol(sce)), sce$bc_id)
sapply(names(cs), function(id) {
  sub <- sce[, cs[[id]]]
  mean(sub$delta > seps[id])
})

# view yield plots including current cutoff
plotYields(sce, which = "A1")
```

`extractClusters` *Extract clusters from a* `SingleCellExperiment`

Description

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

Usage

```
extractClusters(
  x,
  k,
  clusters = NULL,
  as = c("flowSet", "fcs"),
  out_dir = ".",
  verbose = TRUE
)
```

Arguments

`x`	a `SingleCellExperiment`.
`k`	numeric or character string. Specifies the clustering to extract populations from. Must be one of `names(cluster_codes(x))`.
`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 writen 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 & Helena L Crowell <helena.crowell@uzh.ch>

Examples

```
# construct SCE & run clustering
data(PBMC_fs, PBMC_panel, PBMC_md, merging_table)
sce <- prepData(PBMC_fs, PBMC_panel, PBMC_md)
sce <- cluster(sce)

# merge clusters
sce <- mergeClusters(sce, k="meta20", table=merging_table, id="merging_1")
extractClusters(sce, k="merging_1", clusters=c("NK cells", "surface-"))
```

---

filterSCE SingleCellExperiment *filtering*

Description

Filters cells/features from a SingleCellExperiment using conditional statements a la dplyr.

Usage

```
filterSCE(x, ..., k = NULL)
```

Arguments

x	a SingleCellExperiment.
...	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 names(cluster_codes(x)). Defaults to the 1st clustering available.

Value

a SingleCellExperiment.

Author(s)

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

Examples

```
# construct SCE & run clustering
data(PBMC_fs, PBMC_panel, PBMC_md, merging_table)
sce <- prepData(PBMC_fs, PBMC_panel, PBMC_md)
sce <- cluster(sce)

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

# keep only a subset of clusters
filterSCE(sce, 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, sep = "_")
```

Arguments

x	a `flowFrame`.
sep	character string specifying how channel descriptions should be parsed. E.g., if `pData(x)$desc` contains both channel and antigens formatted as, `155Gd_CD73`, descriptions will be split according to `sep` and everything after the first `sep` will be used as the antigen name (here, CD73).

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

Examples

```
# exemplary data with Time, DNA, BC channels, etc.
data(raw_data)
guessPanel(raw_data[[1]])
```

`mergeClusters` *Manual cluster merging*

Description

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

Usage

```
mergeClusters(x, k, table, id, overwrite = FALSE)
```

Arguments

`x`	a `SingleCellExperiment`.
`k`	character string specifying the clustering to merge; valid values are `names(cluster_codes(x))`.
`table`	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.
`overwrite`	logical specifying whether to force overwriting should a clustering with name `id` already exist.

Details

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

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

Value

a `SingleCellExperiment` with newly added cluster codes stored in `cluster_codes(.)$id`.

Author(s)

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

References

Nowicka M, Krieg C, Crowell HL, 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

```
# construct SCE & run clustering
data(PBMC_fs, PBMC_panel, PBMC_md, merging_table)
sce <- prepData(PBMC_fs, PBMC_panel, PBMC_md)
sce <- cluster(sce)

# merge clusters
sce <- mergeClusters(sce,
  k = "meta20",
  id = "merging",
  table = merging_table)

# tabulate manual merging
table(cluster_ids(sce, k = "merging"))

# visualize median type-marker expression
plotExprHeatmap(sce,
  features = "type",
  by = "cluster_id",
  k = "merging",
  bars = TRUE)
```

---

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,
  beads = c("dvs", "beta"),
  dna = c(191, 193),
  k = 500,
  trim = 5,
  remove_beads = TRUE,
  norm_to = NULL,
  assays = c("counts", "exprs"),
  overwrite = TRUE,
  transform = TRUE,
  cofactor = NULL,
  plot = TRUE,
  verbose = TRUE
)
```

Arguments

x	a SingleCellExperiment.
beads	"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.
dna	numeric vector of masses corresponding to DNA channels (only one is required; output scatter plot (see Value section) will be generated using the first matching channel).
k	integer width of the median window used for bead smoothing (affects visualizations only!).
trim	a single non-negative numeric. A *median*+/-trim**mad* rule is applied to preliminary bead populations to remove bead-bead doublets and low signal beads prior to estimating normalization factors.
remove_beads	logical. If TRUE, bead events will be removed from the input SingleCellExperiment and returned as a separate object?
norm_to	a flowFrame or character string specifying an FCS file from which to compute baseline bead intensities, and to which the input data should be normalized to.
assays	lnegth 2 character string specifying which assay data to use; both should be in assayNames(x) and correspond to count- and expression-like data, respectively.
overwrite	logical; should the specified assays (both, when transform = TRUE) be overwritten with the normalized data? If FALSE, normalized counts (and expressions, if transform = TRUE) will be stored in assay(s) normcounts/exprs, respectively.
transform	logical; should normalized counts be arcsinh-transformed with the specified cofactor(s)?
cofactor	numeric cofactor(s) to use for optional arcsinh-transformation when transform = TRUE; single value or a vector with channels as names. If NULL, normCytof will try and access the cofactor(s) stored in int_metadata(x), thus re-using the same transformation applied previsouly.
plot	logical; should bead vs. DNA scatters and smoothed bead intensities before vs. after normalization be included in the output?
verbose	logical; should extra information on progress be reported?

Value

a list of the following SingleCellExperiment...

- data: The filtered input SCE (when remove_beads = TRUE); otherwise, colData columns is_bead and remove indicate whether an event as been identified as a bead or doublet. If overwrite = FALSE, assays normcounts/exprs are added; otherwise, the specified counts/exprs assays are overwritten.
- beads, removed: SCEs containing subsets of events identified as beads and that were removed, respectively. The latter includes bead-cell and cell-cell doublets)

...and ggplot objects:

- scatter: scatter plot of DNA vs. bead intensities with indication of the applied gates
- lines: running-median smoothed bead intensities before and after normalization

Author(s)

Helena L 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)
sce <- prepData(raw_data)

# apply normalization & write normalized data to separate assays
res <- normCytof(sce, beads = "dvs", k = 80, overwrite = FALSE)

ncol(res$beads)   # no. of bead events
ncol(res$removed) # no. of events removed

res$scatter # plot DNA vs. bead intensities including applied gates
res$lines   # plot smoothed bead intensities before vs. after normalization

# filtered SCE now additionally includes
# normalized count & expression data
assayNames(res$data)
```

---

pbMDS *Pseudobulk-level MDS plot*

Description

Pseudobulk-level Multi-Dimensional Scaling (MDS) plot computed on median marker expressions in each sample.

Usage

```
pbMDS(
  x,
  by = c("sample_id", "cluster_id", "both"),
  k = "meta20",
  dims = c(1, 2),
  features = NULL,
  assay = "exprs",
  fun = c("median", "mean", "sum"),
  color_by = switch(by, sample_id = "condition", "cluster_id"),
  label_by = if (by == "sample_id") "sample_id" else NULL,
  shape_by = NULL,
```

```
  size_by = is.null(shape_by),
  pal = if (color_by == "cluster_id") .cluster_cols else NULL
)
```

Arguments

x	a SingleCellExperiment.
by	character string specifying whether to aggregate by sample_id, cluster_id or both.
k	character string specifying which clustering to use when by != "sample_id"; valid values are names(cluster_codes(x)).
dims	two numeric scalars indicating which dimensions to plot.
features	character string specifying which features to include for computation of reduced dimensions; valid values are "type"/"state" for type/state_markers(x) if rowData(x)$marker_class have been specified; a subset of rownames(x); NULL to use all features.
assay	character string specifying which assay data to use; valid values are assayNames(x).
fun	character string specifying which summary statistic to use.
color_by	character string specifying a non-numeric cell metadata column to color by; valid values are names(colData(x)).
label_by	character string specifying a non-numeric cell metadata column to label by; valid values are names(colData(x)).
shape_by	character string specifying a non-numeric cell metadata column to shape by; valid values are names(colData(x)).
size_by	logical specifying whether points should be sized by the number of cells that went into aggregation; i.e., the size of a give sample, cluster or cluster-sample instance.
pal	character vector of colors to use; NULL for default ggplot2 colors.

Value

a ggplot object.

Author(s)

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

References

Nowicka M, Krieg C, Crowell HL, 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)
sce <- prepData(PBMC_fs, PBMC_panel, PBMC_md)
sce <- cluster(sce)

# sample-level pseudobulks
# including state-markers only
pbMDS(sce, by = "sample_id", features = "state")

# cluster-level pseudobulks
# including type-features only
pbMDS(sce, by = "cluster_id", features = "type")

# pseudobulks by cluster-sample
# including all features
pbMDS(sce, by = "both", k = "meta12",
  shape_by = "condition", size_by = TRUE)
```

---

plotAbundances *Population frequencies across samples & clusters*

Description

Plots the relative population abundances of the specified clustering.

Usage

```
plotAbundances(
  x,
  k = "meta20",
  by = c("sample_id", "cluster_id"),
  group_by = "condition",
  shape_by = NULL,
  col_clust = TRUE,
 distance = c("euclidean", "maximum", "manhattan", "canberra", "binary", "minkowski"),
  linkage = c("average", "ward.D", "single", "complete", "mcquitty", "median",
    "centroid", "ward.D2"),
  k_pal = .cluster_cols
)
```

Arguments

x	a SingleCellExperiment.
k	character string specifying which clustering to use; valid values are `names(cluster_codes(x))`.
by	a character string specifying whether to plot frequencies by samples or clusters.

group_by	character string specifying a non-numeric cell metadata columnd to group by (determines the color coding); valid values are `names(colData(x))` other than "sample_id" and "cluster_id".
shape_by	character string specifying a non-numeric cell metadata columnd to shape by; valid values are `names(colData(x))` other than "sample_id" and "cluster_id".
col_clust	for `by = "sample_id"`, specifies whether to hierarchically cluster samples and reorder them accordingly. When `col_clust = FALSE`, samples are ordered according to `levels(x$sample_id)` (or alphabetically, when `x$sample_id` is not a factor).
distance	character string specifying the distance metric to use for sample clustering; passed to `dist`
linkage	character string specifying the agglomeration method to use for sample clustering; passed to `hclust`.
k_pal	character string specifying the cluster color palette; ignored when `by = "cluster_id"`. If less than `nlevels(cluster_ids(x, k))` are supplied, colors will be interpolated via `colorRampPalette`.

Value

a `ggplot` object.

Author(s)

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

References

Nowicka M, Krieg C, Crowell HL, 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

```
# construct SCE & run clustering
data(PBMC_fs, PBMC_panel, PBMC_md)
sce <- prepData(PBMC_fs, PBMC_panel, PBMC_md)
sce <- cluster(sce)

# plot relative population abundances
# by sample & cluster, respectively
plotAbundances(sce, k = "meta12")
plotAbundances(sce, k = "meta8", by = "cluster_id")

# use custom cluster color palette
plotAbundances(sce, k = "meta10",
  k_pal = c("lightgrey", "cornflowerblue", "navy"))
```

`plotClusterExprs` *Plot expression distributions by cluster*

Description

Plots smoothed densities of marker intensities by cluster.

Usage

```
plotClusterExprs(x, k = "meta20", features = "type")
```

Arguments

`x`	a `SingleCellExperiment`.
`k`	character string specifying which clustering to use; valid values are `names(cluster_codes(x))`.
`features`	a character vector specifying which antigens to include; valid values are `"type"/"state"` for `type/state_markers(x)` if `rowData(x)$marker_class` have been specified; a subset of `rownames(x)`; NULL to use all features.

Value

a `ggplot` object.

Author(s)

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

References

Nowicka M, Krieg C, Crowell HL, 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

```
# construct SCE & run clustering
data(PBMC_fs, PBMC_panel, PBMC_md)
sce <- prepData(PBMC_fs, PBMC_panel, PBMC_md)
sce <- cluster(sce)

plotClusterExprs(sce, k = "meta8")
```

plotCodes *tSNE and PCA on SOM codes*

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, k = "meta20", k_pal = .cluster_cols)
```

Arguments

x	a SingleCellExperiment.
k	character string. Specifies the clustering to use for color coding.
k_pal	character string specifying the cluster color palette; If less than nlevels(cluster_ids(x, k)) are supplied, colors will be interpolated via colorRampPalette.

Value

a ggplot object.

Author(s)

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

References

Nowicka M, Krieg C, Crowell HL, 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

```
# construct SCE & run clustering
data(PBMC_fs, PBMC_panel, PBMC_md)
sce <- prepData(PBMC_fs, PBMC_panel, PBMC_md)
sce <- cluster(sce)

plotCodes(sce, k = "meta14")

# use custom cluster color palette
plotCodes(sce, k = "meta12",
  k_pal = c("lightgrey", "cornflowerblue", "navy"))
```

plotCounts *Plot cell counts*

Description

Barplot of the number of cells measured for each sample.

Usage

```
plotCounts(x, group_by = "condition", color_by = group_by, prop = FALSE)
```

Arguments

x	a SingleCellExperiment.
group_by	character string specifying a non-numeric cell metadata column to group by (determines x-axis ticks); valid values are names(colData(x)).
color_by	character string specifying a non-numeric cell metadata column to color by (determines grouping of bars); valid values are names(colData(x)); NULL for no color.
prop	logical specifying whether to plot relative abundances (frequencies) for each group rather than total cell counts; bars will be stacked when prop = TRUE and dodged otherwise.

Value

a ggplot object.

Author(s)

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

References

Nowicka M, Krieg C, Crowell HL, 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)
sce <- prepData(PBMC_fs, PBMC_panel, PBMC_md)

# plot number of cells per sample, colored by condition
plotCounts(sce,
  group_by = "sample_id",
  color_by = "condition")

# same as above, but order by patient
```

```
plotCounts(sce,
  group_by = "patient_id",
  color_by = "condition")

# total number of cell per patient
plotCounts(sce,
  group_by = "patient_id",
  color_by = NULL)

# plot proportion of cells from each patient by condition
plotCounts(sce,
  prop = TRUE,
  group_by = "condition",
  color_by = "patient_id")
```

`plotDiffHeatmap` *Plot differential heatmap*

Description

Heatmaps summarizing differental abundance & differential state testing results.

Usage

```
plotDiffHeatmap(
  x,
  y,
  k = NULL,
  top_n = 20,
  fdr = 0.05,
  lfc = 1,
  all = FALSE,
  sort_by = c("padj", "lfc", "none"),
  y_cols = list(padj = "p_adj", lfc = "logFC", target = "marker_id"),
  assay = "exprs",
  fun = c("median", "mean", "sum"),
  normalize = TRUE,
  col_anno = TRUE,
  row_anno = TRUE,
  hm_pal = NULL,
  fdr_pal = c("lightgrey", "lightgreen"),
  lfc_pal = c("blue3", "white", "red3")
)
```

Arguments

x	a SingleCellExperiment.

y	a SummarizedExperiment 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.
k	character string specifying the clustering in x from which y was obtained. If NULL, plotDiffHeatmap will try and guess it, which will be inaccurate if multiple clusterings share the same levels.
top_n	numeric. Number of top clusters (if type = "DA") or cluster-marker combinations (if type = "DS") to display.
fdr	numeric threshold on adjusted p-values below which results should be retained and considered to be significant.
lfc	numeric threshold on logFCs above which to retain results.
all	logical specifying whether all top_n results should be displayed. If TRUE, fdr,lfc filtering is skipped.
sort_by	character string specifying the y column to sort by; "none" to retain original ordering. Adj. p-values will increase, logFCs will decreasing from top to bottom.
y_cols	named list specifying columns in y that contain adjusted p-values (padj), logFCs (lfc) and, for DS results, feature names (target). When only some y_cols differ from the defaults, specifying only these is sufficient.
assay	character string specifying which assay data to use; valid values are assayNames(x).
fun	character string specifying the function to use as summary statistic for aggregation of assay data.
normalize	logical specifying whether Z-score normalized values should be plotted. If y contains DA analysis results, frequencies will be arcsine-square-root scaled prior to normalization.
col_anno	logical specifying whether to include column annotations for all non-numeric cell metadata variables; or a character vector in names(colData(x)) to include only a subset of annotations. (Only variables that map uniquely to each sample will be included)
row_anno	logical specifying whether to include a row annotation indicating whether cluster (DA) or cluster-marker combinations (DS) are significant, labeled with adjusted p-values, as well as logFCs.
hm_pal	character vector of colors to interpolate for the heatmap. Defaults to brewer.pal's "RdYlBu" for DS, "RdBu" for DA results heatmaps.
fdr_pal, lfc_pal	character vector of colors to use for row annotations • fdr_pallength 2 for (non-)significant at given fdr • lfc_pallength 3 for negative, zero and positive

Value

a Heatmap-class object.

Author(s)

Lukas M Weber & Helena L Crowell <helena.crowell@uzh.ch>

Examples

```
# construct SCE & run clustering
data(PBMC_fs, PBMC_panel, PBMC_md)
sce <- prepData(PBMC_fs, PBMC_panel, PBMC_md)
sce <- cluster(sce, verbose = FALSE)

## differential analysis
library(diffcyt)

# create design & constrast matrix
design <- createDesignMatrix(ei(sce), cols_design=2:3)
contrast <- createContrast(c(0, 1, 0, 0, 0))

# test for
# - differential abundance (DA) of clusters
# - differential states (DS) within clusters

da <- diffcyt(sce, design = design, contrast = contrast,
    analysis_type = "DA", method_DA = "diffcyt-DA-edgeR",
    clustering_to_use = "meta20", verbose = FALSE)

ds <- diffcyt(sce, design = design, contrast = contrast,
    analysis_type = "DS", method_DS = "diffcyt-DS-limma",
    clustering_to_use = "meta20", verbose = FALSE)

# extract result tables
da <- rowData(da$res)
ds <- rowData(ds$res)

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

# visualize results for subset of clusters
sub <- filterSCE(sce, cluster_id %in% seq_len(5), k = "meta20")
plotDiffHeatmap(sub, da, all = TRUE, sort_by = "none")

# visualize results for selected feature
# & include only selected annotation
plotDiffHeatmap(sce["pp38", ], ds, col_anno = "condition", all = TRUE)
```

---

plotDR *Plot reduced dimensions*

Description

Dimension reduction plot colored by expression, cluster, sample or group ID.

Usage

```
plotDR(
  x,
  dr = NULL,
  color_by = "condition",
  facet_by = NULL,
  ncol = NULL,
  assay = "exprs",
  scale = TRUE,
  q = 0.01,
  dims = c(1, 2),
  k_pal = .cluster_cols,
  a_pal = hcl.colors(10, "Viridis")
)
```

Arguments

x	a SingleCellExperiment.
dr	character string specifying which dimension reduction to use. Should be one of reducedDimNames(x); default to the 1st available.
color_by	character string specifying the color coding; valid values are rownames(sce) and names(colData(x)).
facet_by	character string specifying a non-numeric cell metadata column to facet by; valid values are names(colData(x)).
ncol	integer scalar specifying number of facet columns; ignored unless coloring by multiple features without facetting or coloring by a single feature with facetting.
assay	character string specifying which assay data to use when coloring by marker(s); valid values are assayNames(x).
scale	logical specifying whether assay data should be scaled between 0 and 1 using lower (1%) and upper (99%) expression quantiles; ignored if !all(color_by %in% rownames(x)).
q	single numeric in [0,0.5) determining the quantiles to trim when scale = TRUE.
dims	length 2 numeric specifying which dimensions to plot.
k_pal	character string specifying the cluster color palette; ignored when color_by is not one of names(cluster_codes(x)). If less than nlevels(cluster_ids(x, k)) are supplied, colors will be interpolated via colorRampPalette.
a_pal	character string specifying the assay data palette when coloring by feature(s), i.e. all(color_by %in% rownames(x)).

Value

a ggplot object.

Author(s)

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

References

Nowicka M, Krieg C, Crowell HL, 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

```
# construct SCE & run clustering
data(PBMC_fs, PBMC_panel, PBMC_md)
sce <- prepData(PBMC_fs, PBMC_panel, PBMC_md)

# run clustering & dimension reduction
sce <- cluster(sce)
sce <- runDR(sce, dr = "UMAP", cells = 100)

# color by single marker, split by sample
plotDR(sce, color_by = "CD7", facet_by = "sample_id", ncol = 4)

# color by a set of markers using custom color palette
cdx <- grep("CD", rownames(sce), value = TRUE)
plotDR(sce, color_by = cdx, ncol = 4,
  a_pal = rev(hcl.colors(10, "Spectral")))

# color by scaled expression for
# set of markers, split by condition
plotDR(sce,
  scale = TRUE,
  facet_by = "condition",
  color_by = sample(rownames(sce), 4))

# color by 8 metaclusters using custom
# cluster color palette, split by sample
p <- plotDR(sce,
  color_by = "meta8",
  facet_by = "sample_id",
  k_pal = c("lightgrey", "cornflowerblue", "navy"))
p$facet$params$ncol <- 4; p
```

`plotEvents` *Event plot*

Description

Plots normalized barcode intensities for a given barcode.

Usage

```
plotEvents(
  x,
  which = "all",
  assay = "scaled",
  n = 1000,
  out_path = NULL,
  out_name = "event_plot"
)
```

Arguments

x	a SingleCellExperiment.
which	"all", numeric or character specifying which barcode(s) to plot. Valid values are IDs that occur as rownames in the bc_key slot of the input SCE's metadata, or 0 for unassigned events.
assay	character string specifying which assay data slot to use. One of assayNames(x).
n	single numeric specifying the number of events to plot.
out_path	character string. If specified, events plots for all barcodes specified via which will be written to a single PDF file in this location.
out_name	character strings specifying the output's file name when !is.null(out_path); should be provided without(!) file type extension.

Details

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.

Value

a list of ggplot objects.

Author(s)

Helena L 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)
sce <- prepData(sample_ff, by_time = FALSE)
sce <- assignPrelim(sce, sample_key)
plotEvents(sce, which = "D1")
```

---

plotExprHeatmap *Plot expression heatmap*

Description

Heatmap of marker expressions aggregated by sample, cluster, or both; with options to include annotation of cell metadata factors, clustering(s), as well as relative and absolute cell counts.

Usage

```
plotExprHeatmap(
  x,
  features = NULL,
  by = c("sample_id", "cluster_id", "both"),
  k = "meta20",
  m = NULL,
  assay = "exprs",
  fun = c("median", "mean", "sum"),
  scale = c("first", "last", "never"),
  q = 0.01,
  row_anno = TRUE,
  col_anno = TRUE,
  row_clust = TRUE,
  col_clust = TRUE,
  row_dend = TRUE,
  col_dend = TRUE,
  bars = FALSE,
  perc = FALSE,
  bin_anno = FALSE,
  hm_pal = rev(brewer.pal(11, "RdYlBu")),
  k_pal = .cluster_cols,
  m_pal = k_pal,
 distance = c("euclidean", "maximum", "manhattan", "canberra", "binary", "minkowski"),
  linkage = c("average", "ward.D", "single", "complete", "mcquitty", "median",
    "centroid", "ward.D2")
)
```

Arguments

x	a SingleCellExperiment.
features	character string specifying which features to include; valid values are "type"/"state" for type/state_markers(x) if rowData(x)$marker_class have been specified; a subset of rownames(x); NULL to use all features. When by = "both", only 1 feature is allowed.
by	character string specifying whether to aggregate by sample, cluster, both.
k	character string specifying which clustering to use when by != "sample_id"; assay data will be aggregated across these cluster IDs.
m	character string specifying a metaclustering to include as an annotation when by != "sample_id" and row_anno = TRUE.
assay	character string specifying which assay data to use; valid values are assayNames(x).
fun	character string specifying the function to use as summary statistic.
scale	character string specifying the scaling strategy: • "first": scale & trim then aggregate • "last": aggregate then scale & trim • "never": aggregate only If scale != "never", data will be scaled using lower (q%) and upper (1-q%) quantiles as boundaries.
q	single numeric in [0,0.5) determining the quantiles to trim when scale != "never".
row_anno, col_anno	logical specifying whether to include row/column annotations (see details); when one axis corresponds to samples (by != "cluster_id"), this can be a character vector specifying a subset of names(colData(x)) to be included as annotations.
row_clust, col_clust	logical specifying whether rows/columns should be hierarchically clustered and re-ordered accordingly.
row_dend, col_dend	logical specifying whether to include the row/column dendrograms.
bars	logical specifying whether to include a barplot of cell counts per cluster as a right-hand side row annotation.
perc	logical specifying whether to display percentage labels next to bars when bars = TRUE.
bin_anno	logical specifying whether to display values inside bins.
hm_pal	character vector of colors to interpolate for the heatmap.
k_pal, m_pal	character vector of colors to interpolate for cluster annotations when by != "sample_id".
distance	character string specifying the distance metric to use for both row and column hierarchical clustering; passed to Heatmap
linkage	character string specifying the agglomeration method to use for both row and column hierarchical clustering; passed to Heatmap

Details

By default (row/col_anno = TRUE), for axes corresponding to samples (y-axis for by = "sample_id" and x-axis for by = "both"), annotations will be drawn for all non-numeric cell metadata variables. Alternatively, a specific subset of annotations can be included for only a subset of variables by specifying row/col_anno to be a character vector in names(colData(x)) (see examples).

For axes corresponding to clusters (y-axis for by = "cluster_id" and "both"), annotations will be drawn for the specified clustering(s) (arguments k and m).

Value

a Heatmap-class object.

Author(s)

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

References

Nowicka M, Krieg C, Crowell HL, 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)

See Also

plotFreqHeatmap, plotMultiHeatmap

Examples

```
data(PBMC_fs, PBMC_panel, PBMC_md)
sce <- prepData(PBMC_fs, PBMC_panel, PBMC_md)
sce <- cluster(sce)

# median scaled & trimmed expression by cluster
plotExprHeatmap(sce,
  by = "cluster_id", k = "meta8",
  scale = "first", q = 0.05, bars = FALSE)

# scale each marker between 0 and 1
# after aggregation (without trimming)
plotExprHeatmap(sce,
  scale = "last", q = 0,
  bars = TRUE, perc = TRUE,
  hm_pal = hcl.colors(10, "YlGnBu", rev = TRUE))

# raw (un-scaled) median expression by cluster-sample
plotExprHeatmap(sce,
  features = "pp38", by = "both", k = "meta10",
  scale = "never", row_anno = FALSE, bars = FALSE)

# include only subset of samples
sub <- filterSCE(sce,
```

```
  patient_id != "Patient",
  sample_id != "Ref3")

# includes specific annotations &
# split into CDx & all other markers
is_cd <- grepl("CD", rownames(sce))
plotExprHeatmap(sub,
  rownames(sce)[is_cd],
  row_anno = "condition",
  bars = FALSE)
plotExprHeatmap(sub,
  rownames(sce)[!is_cd],
  row_anno = "patient_id",
  bars = FALSE)
```

`plotExprs` *Expression densities*

Description

Plots smoothed densities of marker intensities, with a density curve for each sample ID, and curves colored by a cell metadata variable of interest.

Usage

```
plotExprs(x, features = NULL, color_by = "condition", assay = "exprs")
```

Arguments

`x`	a `SingleCellExperiment`.
`features`	character vector specifying which features to invlude; valid values are `"type"/"state"` for `type/state_markers(x)` if `rowData(x)$marker_class` have been specified; a subset of `rownames(x)`; NULL to use all features.
`color_by`	character string specifying a non-numeric cell metadata column by which to color density curves for each sample; valid values are `names(colData(x))`.
`assay`	character string specifying which assay data to use; valid values are `assayNames(x)`.

Value

a `ggplot` object.

Author(s)

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

References

Nowicka M, Krieg C, Crowell HL, 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)
sce <- prepData(PBMC_fs, PBMC_panel, PBMC_md)
plotExprs(sce)
```

`plotFreqHeatmap` *Cluster frequency heatmap*

Description

Heatmap of relative cluster abundances (frequencies) by sample.

Usage

```
plotFreqHeatmap(
  x,
  k = "meta20",
  m = NULL,
  normalize = TRUE,
  row_anno = TRUE,
  col_anno = TRUE,
  row_clust = TRUE,
  col_clust = TRUE,
  row_dend = TRUE,
  col_dend = TRUE,
  bars = TRUE,
  perc = FALSE,
  hm_pal = rev(brewer.pal(11, "RdBu")),
  k_pal = .cluster_cols,
  m_pal = k_pal
)
```

Arguments

x	a SingleCellExperiment.
k	character string specifying the clustering to use; valid values are `names(cluster_codes(x))`. Cell counts will be computed across these cluster IDs.
m	character string specifying a metaclustering to include as an annotation when `row_anno = TRUE`.

normalize — logical specifying whether to Z-score normalize.

row_anno, col_anno — logical specifying whether to include row/column annotations for clusters/samples; for col_anno, this can be a character vector specifying a subset of names(colData(x)) to be included.

row_clust, col_clust — logical specifying whether rows/columns (clusters/samples) should be hierarchically clustered and re-ordered accordingly.

row_dend, col_dend — logical specifying whether to include row/column dendrograms.

bars — logical specifying whether to include a barplot of cell counts per cluster as a right-hand side row annotation.

perc — logical specifying whether to display percentage labels next to bars when bars = TRUE.

hm_pal — character vector of colors to interpolate for the heatmap.

k_pal, m_pal — character vector of colors to use for cluster and merging row annotations. If less than nlevels(cluster_ids(x, k/m)) values are supplied, colors will be interpolated via colorRampPalette.

Value

a Heatmap-class object.

Author(s)

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

See Also

plotAbundances, plotExprHeatmap, plotMultiHeatmap,

Examples

```
data(PBMC_fs, PBMC_panel, PBMC_md)
sce <- prepData(PBMC_fs, PBMC_panel, PBMC_md)
sce <- cluster(sce)

# complete
plotFreqHeatmap(sce, k = "meta12", m = "meta8")

# minimal
plotFreqHeatmap(sce, k = "meta10",
  normalize = FALSE, bars = FALSE,
  row_anno = FALSE, col_anno = FALSE,
  row_clust = FALSE, col_clust = FALSE)

# customize colors & annotations
plotFreqHeatmap(sce,
  k = "meta7", m = "meta4",
```

```
  col_anno = "condition",
  hm_pal = c("navy", "grey95", "gold"),
  k_pal = hcl.colors(7, "Set 2"),
  m_pal = hcl.colors(4, "Dark 3"))
```

`plotMahal` *Biaxial plot*

Description

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

Usage

```
plotMahal(x, which, assay = "exprs", n = 1000)
```

Arguments

`x`	a `SingleCellExperiment`.
`which`	character string. Specifies which barcode to plot.
`assay`	character string specifying which assay to use.
`n`	numeric. Number of cells to subsample; use NULL to include all.

Value

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

Author(s)

Helena L 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)
sce <- prepData(sample_ff, by_time = FALSE)
sce <- assignPrelim(sce, sample_key)
sce <- estCutoffs(sce)
sce <- applyCutoffs(sce)
plotMahal(sce, which = "B3")
```

plotMultiHeatmap *Multi-panel expression & frequency heatmaps*

Description

Combines expression and frequency heatmaps from plotExprHeatmap and plotFreqHeatmap, respectively, into a HeatmapList.

Usage

```
plotMultiHeatmap(
  x,
  hm1 = "type",
  hm2 = "abundances",
  k = "meta20",
  m = NULL,
  assay = "exprs",
  fun = c("median", "mean", "sum"),
  scale = c("first", ifelse(hm2 == "state", "first", "last")),
  q = c(0.01, ifelse(hm2 == "state", 0.01, 0)),
  normalize = TRUE,
  row_anno = TRUE,
  col_anno = TRUE,
  row_clust = TRUE,
  col_clust = c(TRUE, hm2 == "state"),
  row_dend = TRUE,
  col_dend = c(TRUE, hm2 == "state"),
  bars = FALSE,
  perc = FALSE,
  hm1_pal = rev(brewer.pal(11, "RdYlBu")),
 hm2_pal = if (isTRUE(hm2 == "abundances")) rev(brewer.pal(11, "PuOr")) else hm1_pal,
  k_pal = .cluster_cols,
  m_pal = k_pal,
 distance = c("euclidean", "maximum", "manhattan", "canberra", "binary", "minkowski"),
  linkage = c("average", "ward.D", "single", "complete", "mcquitty", "median",
    "centroid", "ward.D2")
)
```

Arguments

x	a SingleCellExperiment.
hm1	character string specifying which features to include in the 1st heatmap; valid values are "type"/"state" for type/state_markers(x) if rowData(x)$marker_class have been specified; a subset of rownames(x); NULL to use all features; and FALSE to omit the 1st heatmap altogether.
hm2	character string. Specifies the right-hand side heatmap. One of:

- "abundances": cluster frequencies across samples
- "state": median 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 specifying which; valid values are names(cluster_codes(x)).

m — character string specifying a metaclustering to include as an annotation when row_anno = TRUE.

assay — character string specifying which assay data to use; valid values are assayNames(x).

fun — character string specifying the function to use as summary statistic.

scale — character string specifying the scaling strategy; for expression heatmaps (see plotExprHeatmap).

q — single numeric in [0,1) determining the quantiles to trim when scale != "never".

normalize — logical specifying whether to Z-score normalize cluster frequencies across samples; see plotFreqHeatmap.

row_anno, col_anno — logical specifying whether to include row/column annotations for cell metadata variables and clustering(s); see plotExprHeatmap and plotFreqHeatmap.

row_clust, col_clust — logical specifying whether rows/columns should be hierarchically clustered and re-ordered accordingly.

row_dend, col_dend — logical specifying whether to include the row/column dendrograms.

bars — logical specifying whether to include a barplot of cell counts per cluster as a right-hand side row annotation.

perc — logical specifying whether to display percentage labels next to bars when bars = TRUE.

hm1_pal, hm2_pal — character vector of colors to interpolate for each heatmap.

k_pal, m_pal — character vector of colors to use for cluster and merging row annotations. If less than nlevels(cluster_ids(x, k/m)) values are supplied, colors will be interpolated via colorRampPalette.

distance — character string specifying the distance metric to use in dist for hierarchical clustering.

linkage — character string specifying the agglomeration method to use in hclust for hierarchical clustering.

Details

In its 1st panel, plotMultiHeatmap will display (scaled) type-marker expressions aggregated by cluster (across all samples). Depending on argument hm2, the 2nd panel will contain one of:

hm2 = "abundances" relataive cluster abundances by cluster & sample

hm2 = "state" aggregated (scaled) state-marker expressions by cluster (across all samples; analogous to panel 1)

hm2 %in% rownames(x) aggregated (scaled) marker expressions by cluster & sample

Value

a HeatmapList-class object.

Author(s)

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

References

Nowicka M, Krieg C, Crowell HL, 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)

See Also

plotAbundances, plotExprHeatmap, plotFreqHeatmap

Examples

```
# construct SCE & run clustering
data(PBMC_fs, PBMC_panel, PBMC_md)
sce <- prepData(PBMC_fs, PBMC_panel, PBMC_md)
sce <- cluster(sce)

# state-markers + cluster frequencies
plotMultiHeatmap(sce,
  hm1 = "state", hm2 = "abundances",
  bars = TRUE, perc = TRUE)

# type-markers + marker of interest
plotMultiHeatmap(sce, hm2 = "pp38", k = "meta12", m = "meta8")

# both, type- & state-markers
plotMultiHeatmap(sce, hm2 = "state")

# plot markers of interest side-by-side
# without left-hand side heatmap
plotMultiHeatmap(sce, k = "meta10",
  hm1 = NULL, hm2 = c("pS6", "pNFkB", "pBtk"),
  row_anno = FALSE, hm2_pal = c("white", "black"))
```

---

plotNRS *Plot non-redundancy scores*

Description

Plots non-redundancy scores (NRS) by feature in decreasing order of average NRS across samples.

Usage

```
plotNRS(x, features = NULL, color_by = "condition", assay = "exprs")
```

Arguments

x	a SingleCellExperiment.
features	a character vector specifying which antigens to use for clustering; valid values are "type"/"state" for type/state_markers(x) if rowData(x)$marker_class have been specified; a subset of rownames(x); NULL to use all features.
color_by	character string specifying the color coding; valid values are namescolData(x)).
assay	character string specifying which assay data to use; valid values are assayNames(x).

Value

a ggplot object.

Author(s)

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

References

Nowicka M, Krieg C, Crowell HL, 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)
sce <- prepData(PBMC_fs, PBMC_panel, PBMC_md)

plotNRS(sce, features = NULL)   # default: all markers
plotNRS(sce, features = "type") # type-markers only
```

---

plotPbExprs *Pseudobulk-level boxplot*

Description

Boxplot of aggregated marker data by sample or cluster, optionally colored and faceted by non-numeric cell metadata variables of interest.

Usage

```
plotPbExprs(
  x,
  k = "meta20",
  features = "state",
  assay = "exprs",
  fun = c("median", "mean", "sum"),
  facet_by = c("antigen", "cluster_id"),
  color_by = "condition",
  group_by = color_by,
  shape_by = NULL,
  size_by = FALSE,
  geom = c("both", "points", "boxes"),
  jitter = TRUE,
  ncol = NULL
)
```

Arguments

x	a SingleCellExperiment{SingleCellExperiment}.
k	character string specifying which clustering to use; values values are names(cluster_codes(x)). Ignored if facet_by = "antigen".
features	character vector specifying which features to include; valid values are "type"/"state" for type/state_markers(x) if rowData(x)$marker_class have been specified; a subset of rownames(x); NULL to use all features.
assay	character string specifying which assay data to use; valid values are assayNames(x).
fun	character string specifying the summary statistic to use.
facet_by	"antigen" or "cluster_id"; the latter requires having run cluster.
color_by, group_by, shape_by	character string specifying a non-numeric cell metadata variable to color, group and shape by, respectively; valid values are names(colData(x)) and names(cluster_codes(x)) if cluster has been run.
size_by	logical specifying whether to scale point sizes by the number of cells in a given sample or cluster-sample instance; ignored when geom = "boxes".
geom	character string specifying whether to include only points, boxplots or both.
jitter	logical specifying whether to use position_jitterdodge in geom_point when geom != "boxes".
ncol	integer scalar specifying number of facet columns.

Value

a ggplot object.

Author(s)

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

References

Nowicka M, Krieg C, Crowell HL, 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

```
# construct SCE
data(PBMC_fs, PBMC_panel, PBMC_md)
sce <- prepData(PBMC_fs, PBMC_panel, PBMC_md)
sce <- cluster(sce, verbose = FALSE)

# plot median expressions by sample & condition
# ...split by marker
plotPbExprs(sce,
  shape_by = "patient_id",
  features = sample(rownames(sce), 6))

# ...split by cluster
plotPbExprs(sce, facet_by = "cluster_id", k = "meta6")

# plot median type-marker expressions by sample & cluster
plotPbExprs(sce, feature = "type", k = "meta6",
  facet_by = "antigen", group_by = "cluster_id", color_by = "sample_id",
  size_by = TRUE, geom = "points", jitter = FALSE, ncol = 5)

# plot median state-marker expressions
# by sample & cluster, split by condition
plotPbExprs(sce, k = "meta6", facet_by = "antigen",
  group_by = "cluster_id", color_by = "condition", ncol = 7)
```

---

plotScatter *Scatter plot*

Description

Bivariate scatter plots including visualization of (group-specific) gates, their boundaries and percentage of selected cells.

Usage

```
plotScatter(
  x,
  chs,
  color_by = NULL,
  facet_by = NULL,
  bins = 100,
  assay = "exprs",
```

```
  label = c("target", "channel", "both"),
  zeros = FALSE,
  k_pal = .cluster_cols
)
```

Arguments

x	a SingleCellExperiment.
chs	character string pecifying which channels to plot. Valid values are antigens: rownames(x), channel names: channels(x) or non-mass channels stored in names([int_]colData(x)), and should correspond to numeric variables.
color_by	character string specifying a cell metadata column to color by; valid values are names(colData(x)), names(int_colData(x)); names(cluster_codes(x)) (if cluster has been run); or NULL to color by density.
facet_by	character string specifying a non-numeric cell metadata column to facet by; valid values are names(colData(x)). When length(chs) == 1, 2 facetting variables may be provided, otherwise 1 only.
bins	numeric of length 1 giving the number of bins for geom_hex when coloring by density.
assay	character string specifying which assay data to use. Should be one of assayNames(x).
label	character string specifying axis labels should include antigen targets, channel names, or a concatenation of both.
zeros	logical specifying whether to include 0 values.
k_pal	character string specifying the cluster color palette; ignored when color_by is not one of names(cluster_codes(x)). If less than nlevels(cluster_ids(x, k)) are supplied, colors will be interpolated via colorRampPalette.

Value

a ggplot object.

Author(s)

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

Examples

```
data(raw_data)
sce <- prepData(raw_data)

dna_chs <- c("DNA1", "DNA2")
plotScatter(sce, dna_chs, label = "both")

plotScatter(sce,
  chs = sample(rownames(sce), 4),
  color_by = "sample_id")

sce <- prepData(sample_ff)
```

```
ids <- sample(rownames(sample_key), 3)
sce <- assignPrelim(sce, sample_key[ids, ])
sce <- sce[, sce$bc_id %in% ids]

chs <- sample(rownames(sce), 5)
plotScatter(sce, chs, color_by = "bc_id")
plotScatter(sce, chs, color_by = "delta")
```

`plotSpillmat` *Spillover matrix heatmap*

Description

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

Usage

```
plotSpillmat(
  x,
  sm = NULL,
  anno = TRUE,
  isotope_list = CATALYST::isotope_list,
  hm_pal = c("white", "lightcoral", "red2", "darkred"),
  anno_col = "black"
)
```

Arguments

x	a SingleCellExperiment.
sm	spillover matrix to visualize. If NULL, plotSpillmat will try and access metadata(x)$spillover_matr
anno	logical. If TRUE (default), spill percentages are shown inside bins and rows are annotated with the total amount of spill received.
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.
hm_pal	character vector of colors to interpolate.
anno_col	character string specifying the color to use for bin annotations.

Value

a ggplot2-object showing estimated spill percentages as a heatmap with colors ramped to the highest spillover value present.

Author(s)

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

Examples

```
# get single-stained control samples & construct SCE
data(ss_exp)
sce <- prepData(ss_exp)

# debarcode single-positive populations
bc_ms <- c(139, 141:156, 158:176)
sce <- assignPrelim(sce, bc_ms, verbose = FALSE)
sce <- applyCutoffs(estCutoffs(sce))

# estimate & visualize spillover matrix
sce <- computeSpillmat(sce)
plotSpillmat(sce)
```

`plotYields` *Yield plot*

Description

Plots the distribution of barcode separations and yields upon debarcoding as a function of separation cutoffs. If available, currently used separation cutoffs as well as their resulting yields will be indicated in the plot.

Usage

```
plotYields(x, which = 0, out_path = NULL, out_name = "yield_plot")
```

Arguments

`x`	a `SingleCellExperiment`.
`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, yields plots for all barcodes specified via `which` will be written to a single PDF file in this location.
`out_name`	character strings specifying the output's file name when `!is.null(out_path)`; should be provided without(!) file type extension.

Details

The overall yield that will be achieved upon application of the specified set of separation cutoffs is indicated in the summary plot. Respective separation thresholds and their resulting yields are included in each barcode's plot. The separation cutoff value should be chosen such that it appropriately balances confidence in barcode assignment and cell yield.

Value

a list of ggplot objects.

Author(s)

Helena L 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

```
# construct SCE & apply arcsinh-transformation
data(sample_ff, sample_key)
sce <- prepData(sample_ff)

# deconvolute samples & estimate separation cutoffs
sce <- assignPrelim(sce, sample_key)
sce <- estCutoffs(sce)

# all barcodes summary plot
plotYields(sce, which = 0)

# plot for specific sample
plotYields(sce, which = "C1")
```

---

prepData *Data preparation*

Description

Data preparation

Usage

```
prepData(
  x,
  panel = NULL,
  md = NULL,
  features = NULL,
  transform = TRUE,
  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")),
```

```
  by_time = TRUE,
  FACS = FALSE,
  fix_chs = c("common", "all"),
  ...
)
```

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"). If 'panel' is unspecified, it will be constructed from the first input sample via guessPanel.
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 • patient_id: the patient ID • condition: brief sample description (e.g. reference/stimulated, healthy/diseased) If 'md' is unspecified, the flowFrame/Set identifier(s) will be used as sample IDs with no additional metadata factors.
features	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.
transform	logical. Specifies whether an arcsinh-transformation with cofactor cofactor should be performed, in which case expression values (transformed counts) will be stored in assay(x, "exprs").
cofactor	numeric cofactor(s) to use for optional arcsinh-transformation when transform = TRUE; single value or a vector with channels as names.
panel_cols	a names list specifying the panel column names that contain channel names, targeted protein markers, and (optionally) marker classes. When only some panel_cols deviate from the defaults, specifying only these is sufficient.
md_cols	a named list specifying the column names of md that contain the FCS file names, sample IDs, and factors of interest (batch, condition, treatment etc.). When only some md_cols deviate from the defaults, specifying only these is sufficient.
by_time	logical; should samples be ordered by acquisition time? Ignored if !is.null(md) in which case samples will be ordered as they are listed in md[[md_cols$file]]. (see details)
FACS	logical; is this FACS / flow cytometry data? By default, prepData moves non-mass channels to the output SCE's int_colData; FACS = TRUE assures that all channels are kept as assay data. If FALSE, prepData will try and access the input flowFrame/Set's "$CYT" descriptor (keyword(., "$CYT")) to determine the data type; this may be inaccurate for some cytometer descriptors.

fix_chs	specifies the strategy to use in case of panel discrepancies. "common" will retain only channels present in all frames/FCS files; "all" will retain the union of channels across samples. In the latter case, a logical matrix with rows = channels and columns = samples will be stored under metadata slot chs_by_fcs specifying which channels were/n't (FALSE/TRUE) measured in which samples.
...	additional arguments passed to read.FCS. E.g., channel_alias in case of panel discrepancies between frames/ FCS files. By default, transformation = truncate_max_range = FALSE.

Details

By default, non-mass channels (e.g., time, event lengths) will be removed from the output SCE's assay data and instead stored in the object's internal cell metadata (int_colData) to assure these data are not subject to transformations or other computations applied to the assay data.

For more than 1 sample, prepData will concatenate cells into a single SingleCellExperiment object. Note that cells will hereby be order by "Time", regardless of whether by_time = TRUE or FALSE. Instead, by_time determines the sample (not cell!) order; i.e., whether samples should be kept in their original order, or should be re-ordered according to their acquision time stored in keyword(flowSet, "$BTIM").

When a metadata table is specified (i.e. !is.null(md)), argument by_time will be ignored and sample ordering is instead determined by md[[md_cols$file]].

Value

a SingleCellExperiment.

Author(s)

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

Examples

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

# channel-specific transformation
cf <- sample(seq_len(10)[-1], nrow(PBMC_panel), TRUE)
names(cf) <- PBMC_panel$fcs_colname
sce <- prepData(PBMC_fs, cofactor = cf)
int_metadata(sce)$cofactor

# input has different name for "condition"
md <- PBMC_md
m <- match("condition", names(md))
colnames(md)[m] <- "treatment"

# add additional factor variable batch ID
md$batch_id <- sample(c("A", "B"), nrow(md), TRUE)

# specify 'md_cols' that differ from defaults
```

```
factors <- list(factors = c("treatment", "batch_id"))
ei(prepData(PBMC_fs, PBMC_panel, md, md_cols = factors))

# without panel & metadata tables
sce <- prepData(raw_data)

# 'flowFrame' identifiers are used as sample IDs
levels(sce$sample_id)

# panel was guess with 'guessPanel';
# non-mass channels are set to marker class "none"
rowData(sce)
```

---

runDR *Dimension reduction*

Description

Wrapper around dimension reduction methods available through scater, with optional subsampling of cells per each sample.

Usage

```
runDR(
  x,
  dr = c("UMAP", "TSNE", "PCA", "MDS", "DiffusionMap"),
  cells = NULL,
  features = "type",
  assay = "exprs",
  ...
)
```

Arguments

x	a SingleCellExperiment.
dr	character string specifying which dimension reduction to use.
cells	single numeric specifying the maximal number of cells per sample to use for dimension reduction; NULL for all cells.
features	a character vector specifying which antigens to use for dimension reduction; valid values are "type"/"state" for type/state_markers(x) if rowData(x)$marker_class have been specified; a subset of rownames(x); NULL to use all features.
assay	character string specifying which assay data to use for dimension reduction; valid values are assayNames(x).
...	optional arguments for dimension reduction; passed to runUMAP, runTSNE, runPCA, runMDS and runDiffusionMap, respecttively. See ?"scater-red-dim-args" for details.

Value

a ggplot object.

Author(s)

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

References

Nowicka M, Krieg C, Crowell HL, 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

```
# construct SCE
data(PBMC_fs, PBMC_panel, PBMC_md)
sce <- prepData(PBMC_fs, PBMC_panel, PBMC_md)

# run UMAP on <= 200 cells per sample
sce <- runDR(sce, features = type_markers(sce), cells = 100)
```

SCE-accessors *SingleCellExperiment accessors*

Description

Various wrappers to conviniently access slots in a SingleCellExperiment created with prepData, and that are used frequently during differential analysis.

Usage

```
## S4 method for signature 'SingleCellExperiment'
ei(x)

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

## S4 method for signature 'SingleCellExperiment'
channels(x)

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

## S4 method for signature 'SingleCellExperiment'
type_markers(x)
```

```
## S4 method for signature 'SingleCellExperiment'
state_markers(x)

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

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

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

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

## S4 method for signature 'SingleCellExperiment'
delta_area(x)
```

Arguments

x	a SingleCellExperiment.
k	character string specifying the clustering to extract. Valid values are `names(cluster_codes(x))`.

Value

`ei` extracts the experimental design table.

`n_cells` extracts the number of events measured per sample.

`channels` extracts the original FCS file's channel names.

`marker_classes` extracts marker class assignments ("type", "state", "none").

`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_ids` extracts the numeric vector of cluster IDs as inferred by FlowSOM.

`cluster_codes` extracts a `data.frame` containing cluster codes for the FlowSOM clustering, the ConsensusClusterPlus metaclustering, and all mergings done through mergeClusters.

`delta_area` extracts the delta area plot stored in the SCE's `metadata` by cluster

Author(s)

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

Examples

```
# construct SCE & run clustering
data(PBMC_fs, PBMC_panel, PBMC_md)
sce <- prepData(PBMC_fs, PBMC_panel, PBMC_md)
sce <- cluster(sce)
```

```
# view experimental design table
ei(sce)

# quick-access sample & cluster assignments
plot(table(sample_ids(sce)))
plot(table(cluster_ids(sce)))

# access specific clustering resolution
table(cluster_ids(sce, k = "meta8"))

# access marker information
channels(sce)
marker_classes(sce)
type_markers(sce)
state_markers(sce)

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

# view delta area plot (relative change in area
# under CDF curve vs. the number of clusters 'k')
delta_area(sce)
```

`sce2fcs` *SCE to* `flowFrame/Set`

Description

If `split_by = NULL`, the input SCE is converted to a `flowFrame`. Otherwise, it is split into a `flowSet` by the specified `colData` column. Any cell metadata (`colData`) and dimension reductions available in the SCE may be dropped or propagated to the output.

Usage

```
sce2fcs(x, split_by = NULL, keep_cd = FALSE, keep_dr = FALSE, assay = "counts")
```

Arguments

`x`	a `SingleCellExperiment`.
`split_by`	NULL or a character string specifying a `colData(x)` column to split by.
`keep_cd, keep_dr`	logials specifying whether cell metadata (stored in `colData(x)`) and dimension reductions (stored in `reducedDims(x)`), respectively, should be kept or dropped.
`assay`	a character string specifying which assay data to use; valid values are `assayNames(x)`. When writing out FCS files, this should correspond to count-like data!

Value

a flowFrame if split_by = NULL; otherwise a flowSet.

Author(s)

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

Examples

```
# PREPROCESSING
data(sample_ff, sample_key)
sce <- prepData(sample_ff, by_time = FALSE)
sce <- assignPrelim(sce, sample_key, verbose = FALSE)

# split SCE by barcode population
fs <- sce2fcs(sce, split_by = "bc_id")

# do some spot checks
library(flowCore)
library(SingleCellExperiment)

length(fs) == nrow(sample_key)
all(fsApply(fs, nrow)[, 1] == table(sce$bc_id))
identical(t(exprs(fs[[1]])), assay(sce, "exprs")[, sce$bc_id == "A1"])

# DIFFERENTIAL ANALYSIS
data(PBMC_fs, PBMC_panel, PBMC_md)
sce <- prepData(PBMC_fs, PBMC_panel, PBMC_md)
sce <- cluster(sce, verbose = FALSE)

# split by 20 metacluster populations
sce$meta20 <- cluster_ids(sce, "meta20")
fs <- sce2fcs(sce, split_by = "meta20", assay = "exprs")
all(fsApply(fs, nrow)[, 1] == table(sce$meta20))
```

Index

adaptSpillmat, 3
applyCutoffs, 4
assignPrelim, 5

BuildSOM, *9*

channels *(*SCE-accessors*)*, 57
channels,SingleCellExperiment-method *(*SCE-accessors*)*, 57
clrDR, 6
cluster, 9, *48*, *50*, *58*
cluster_codes *(*SCE-accessors*)*, 57
cluster_codes,SingleCellExperiment-method *(*SCE-accessors*)*, 57
cluster_ids *(*SCE-accessors*)*, 57
cluster_ids,SingleCellExperiment,character-method *(*SCE-accessors*)*, 57
cluster_ids,SingleCellExperiment,missing-method *(*SCE-accessors*)*, 57
colorRampPalette, *27*, *29*, *34*, *42*, *45*, *50*
compCytof, *3*, 11
computeSpillmat, 13
ConsensusClusterPlus, *9*, *58*

data, 15
delta_area *(*SCE-accessors*)*, 57
delta_area,SingleCellExperiment-method *(*SCE-accessors*)*, 57
diffcyt, *32*
dist, *27*, *45*

ei *(*SCE-accessors*)*, 57
ei,SingleCellExperiment-method *(*SCE-accessors*)*, 57
estCutoffs, 16
extractClusters, 18

filterSCE, 19
flowFrame, *12*, *15*, *20*, *23*, *59*, *60*
flowSet, *15*, *59*, *60*
FlowSOM, *58*

geom_hex, *50*
ggplot, *10*, *28*, *30*, *40*
guessPanel, 20, *54*

hclust, *27*, *45*
Heatmap, *38*
HeatmapList, *44*

identifier, *54*
isotope_list, *3*, *12*, *14*, *51*
isotope_list *(*data*)*, 15

marker_classes *(*SCE-accessors*)*, 57
marker_classes,SingleCellExperiment-method *(*SCE-accessors*)*, 57
mergeClusters, 21, *58*
merging_table *(*data*)*, 15
mg_cells *(*data*)*, 15

n_cells *(*SCE-accessors*)*, 57
n_cells,SingleCellExperiment-method *(*SCE-accessors*)*, 57
normCytof, 22

PBMC_fs *(*data*)*, 15
PBMC_md *(*data*)*, 15
PBMC_panel *(*data*)*, 15
pbMDS, 24
plotAbundances, *21*, 26, *42*, *46*
plotClusterExprs, 28
plotCodes, 29
plotCounts, 30
plotDiffHeatmap, 31
plotDR, 33
plotEvents, 35
plotExprHeatmap, 37, *42*, *44–46*
plotExprs, 40
plotFreqHeatmap, *39*, 41, *44–46*
plotMahal, 43
plotMultiHeatmap, *21*, *39*, *42*, 44
plotNRS, 46

plotPbExprs, 47
plotScatter, 49
plotSpillmat, 51
plotYields, 52
prepData, 53, *57*

raw_data *(*data*)*, 15
read.FCS, *55*
runDiffusionMap, *56*
runDR, 56
runMDS, *56*
runPCA, *56*
runTSNE, *56*
runUMAP, *56*

sample_ff *(*data*)*, 15
sample_ids *(*SCE-accessors*)*, 57
sample_ids,SingleCellExperiment-method
*(*SCE-accessors*)*, 57
sample_key *(*data*)*, 15
SCE-accessors, 57
sce2fcs, 59
SingleCellExperiment, *4*, *6, 7*, *9*, *11*, *13*, *16*, *18, 19*, *21*, *23*, *25, 26*, *28–31*, *34*, *36*, *38*, *40, 41*, *43, 44*, *47, 48*, *50–52*, *55–59*
ss_exp *(*data*)*, 15
state_markers *(*SCE-accessors*)*, 57
state_markers,SingleCellExperiment-method
*(*SCE-accessors*)*, 57

testDA_edgeR, *32*
testDA_GLMM, *32*
testDA_voom, *32*
testDS_limma, *32*
testDS_LMM, *32*
type_markers *(*SCE-accessors*)*, 57
type_markers,SingleCellExperiment-method
*(*SCE-accessors*)*, 57