

# Package ‘cytofQC’

November 21, 2024

**Type** Package

**Title** Labels normalized cells for CyTOF data and assigns probabilities for each label

**Version** 1.7.0

**Description** cytofQC is a package for initial cleaning of CyTOF data. It uses a semi-supervised approach for labeling cells with their most likely data type (bead, doublet, debris, dead) and the probability that they belong to each label type. This package does not remove data from the dataset, but provides labels and information to aid the data user in cleaning their data. Our algorithm is able to distinguish between doublets and large cells.

**License** Artistic-2.0

**URL** <https://github.com/jillbo1000/cytofQC>

**BugReports** <https://github.com/jillbo1000/cytofQC/issues>

**biocViews** Software, SingleCell, Annotation

**Encoding** UTF-8

**LazyData** false

**Imports** CATALYST, flowCore, e1071, EZtune, gbm, ggplot2, hrbrthemes, matrixStats, randomForest, rmarkdown, SingleCellExperiment, stats, SummarizedExperiment, ssc, S4Vectors, graphics, methods

**RoxygenNote** 7.2.3

**Suggests** gridExtra, knitr, RColorBrewer, testthat, uwot

**VignetteBuilder** knitr

**git\_url** <https://git.bioconductor.org/packages/cytofQC>

**git\_branch** devel

**git\_last\_commit** c4c778e

**git\_last\_commit\_date** 2024-10-29

**Repository** Bioconductor 3.21

**Date/Publication** 2024-11-21

**Author** Jill Lundell [aut, cre] (ORCID:

<https://orcid.org/0000-0002-6048-4700>),

Kelly Street [aut] (ORCID: <https://orcid.org/0000-0001-6379-5013>)>

**Maintainer** Jill Lundell <jflundell@gmail.com>

## Contents

cytofQC-package	2
cytofHist	3
cytofQCreport	4
gbmLabel	4
initialBead	6
initialDead	7
initialDebris	8
initialDoublet	9
initialGuess	10
labelQC	11
modelData	12
plotInitialGuess	13
readCytof	14
rfLabel	16
s3vmLabel	17
scores	19
svmLabel	20
<b>Index</b>	<b>22</b>

---

cytofQC-package	<i>cytofQC: Event labeling for quality control of CyTOF data</i>
-----------------	--

---

## Description

Labels observations in a CyTOF dataset as a cell, gdpZero (zero for at least one Gaussian parameter), bead, debris, doublet, or dead cell.

## Details

Data from an fcs file are read directly into a [SingleCellExperiment](#) using the `readCytof` function. The data can be labeled with a single function, `labelQC`, which can be customized. Labeling can also be done using a set of other functions that first select a set of events that clearly look like the event type being modeled and then use those events to train a statistical learning model that can identify the event type. These functions are discussed and demonstrated in the vignette.

A plotting function called `cytofHist` is included that makes assessing the characteristic of the data and labeling easy.

The package also includes a function called `cytofQCreport` that generates a report of the labeling and can generate a umap created with the QC variables and colored by event label.

```

Package: cytofQC
Type: Package
Version: 0.99.3
Date: 2022-11-11
License: Artistic-2.0

```

**Author(s)**

Maintainer: Jill Lundell <jflundell@gmail.com>

Authors: J. Lundell, K. Street

---

cytofHist

*Returns histogram for grouped data*

---

**Description**

Returns histogram for grouped data

**Usage**

```
cytofHist(x, group, type = c("count", "density"), na.rm = FALSE, title = NULL)
```

**Arguments**

x	Numeric vector of values that will be plotted.
group	A vector that contains the grouping variable. It can be a numeric, factor, or character vector.
type	Either "count" or "density". The "count" selection keeps the groups on the same scale. The "density" option will over emphasize the group with the fewest observations. This is helpful when identifying where certain subgroups are relative to the majority of the data.
na.rm	TRUE if NAs should be removed prior to plotting. FALSE if they should remain. The NAs will be plotted as a separate group if they are not removed.
title	Optional title for the plot

**Value**

A ggplot2 histogram.

**Examples**

```
data("raw_data", package = "CATALYST")
sce <- readCytof(raw_data, beads = 'Beads', viability = c('cisPt1','cisPt2'))
sce <- labelQC(sce)
cytofHist(scores(sce, 'bead'), label(sce))
```

---

cytofQCreport	<i>Generate a cytofQC report</i>
---------------	----------------------------------

---

### Description

Generate a cytofQC report

### Usage

```
cytofQCreport(x, outDir, sampName, runUMAP = TRUE, ...)
```

### Arguments

x	A SingleCellExperiment object generated by <a href="#">labelQC</a> .
outDir	The output directory (currently required).
sampName	Basename of the output HTML file (if not provided, same as outDir).
runUMAP	Logical value indicating whether or not to include a UMAP plot in the report. This plot can be beneficial for diagnostic purposes, but is time-consuming to generate.
...	Additions arguments that may be passed to the function.

### Value

If successful, returns TRUE silently and generates the specified QC report.

### Examples

```
data("raw_data", package = "CATALYST")
x <- readCytobf(raw_data, beads = "Beads", viability = c("cisPt1", "cisPt2"))
x <- labelQC(x, n = 500)
tmp <- tempdir()
cytofQCreport(x, tmp, 'example')
```

---

gbmLabel	<i>Returns the final label assignments for a parameter using a gradient boosting machine</i>
----------	--

---

### Description

Returns the final label assignments for a parameter using a gradient boosting machine

**Usage**

```
gbmLabel(
  x,
  type = c("bead", "doublet", "debris", "dead"),
  loss = c("auc", "class"),
  n = 4000,
  standardize = TRUE
)
```

**Arguments**

x	A SingleCellExperiment created with <a href="#">readCytof</a> with the scores and initial columns filled out for the event type of interest.
type	Identifies the type of label that is being modeled. Must be 'bead', 'doublet', 'debris', or 'dead'.
loss	Specifies the type of loss used to tune the GBM. Can be either "auc" for the area under the curve or "class" for classification error.
n	number of observations in training dataset.
standardize	Indicates if the data should be standardized. Because the data are on different scales, it should be standardized for this analysis.

**Details**

gbmLabel uses a gradient boosting machine to compute the final labels for the specified parameter type (bead, doublet, debris, or dead). This step cannot be completed until the corresponding initialization function (initialBead, initialDebris, initialDoublet, or initialDead) is done on the SingleCellExperiment created by readCytof. The gbm is tuned using [eztune](#) and then predicted values are computed for all of the events in x. If the predicted probability for the label type is greater than 0.5, the label is changed to the specified type. However, if an observation already has a label other than 'cell' in the label variable, it will not be changed. The predicted probabilities for all of the observations are stored in the variable associated with that type in the probs object of x for further analysis. Thus, it is possible to have a probability greater than 0.5 for 'debris' but still have a label of 'bead' if an observation was classified as a bead prior to classifying the debris.

**Value**

An updated SingleCellExperiment is returned with the labels for the parameter of interest (bead, doublet, debris, or dead) added to the label object of the SingleCellExperiment and the probabilities for the event type added to the probs object of the SingleCellExperiment.

**Examples**

```
data("raw_data", package = "CATALYST")
sce <- readCytof(raw_data, beads = "Beads", viability = c("cisPt1", "cisPt2"))
sce <- initialBead(sce)
sce <- gbmLabel(sce, type = "bead", loss = "auc")
head(probs(sce))
table(label(sce))
```

---

initialBead	<i>Preliminary bead classification</i>
-------------	--

---

### Description

Preliminary bead classification

### Usage

```
initialBead(x)
```

### Arguments

x                    A SingleCellExperiment created with [readCytof](#).

### Details

The beads are typically the first cell classification that is done. The different event types are labeled iteratively so the `labels` vector in the `colData` will contain all of the labels and probabilities computed up to this point. Only events that have a "cell" label can be assigned an initial event classification of "bead". This function computes a score that assesses how much an event looks like a bead and then fits a mixture model to assign each event a class of 1 for a bead, -1 for an event that is not a bead, or 0 for undetermined or previously assigned to a different event type. The score is recorded in the `score` object in the `colData` and the initial classification is recorded in the `initial` part of the `colData`.

Each bead channel should classify into two fairly clear groups where one is the beads and the other is non-beads. A histogram of the bead score should show a clear, small peak that represents the beads.

### Value

A `SingleCellExperiment` that contains the bead score and the bead designation for each event. This information is stored in the `score` and `initial` objects in the `colData` for the `SingleCellExperiment`.

### Examples

```
data("raw_data", package = "CATALYST")
sce <- readCytof(raw_data, beads = 'Beads', viability = c('cisPt1','cisPt2'))
sce <- initialBead(sce)
head(scores(sce))
head(initial(sce))
```

---

initialDead	<i>Preliminary viability classification</i>
-------------	---

---

### Description

Preliminary viability classification

### Usage

```
initialDead(x, dna = FALSE, standardize = TRUE)
```

### Arguments

x	A SingleCellExperiment created with <a href="#">readCytof</a> .
dna	If TRUE, the DNA will be used to determine viability.
standardize	A value of TRUE will use compute the doublet score using standardized data. The raw data will be used to compute the score if FALSE. It is highly recommended that the data are standardized prior to computing the score because the variables are on different scales.

### Details

The beads are typically the first cell classification that is done because their identification is straightforward. Debris is typically classified after the beads and doublets classified after the debris. After the doublets are classified, the permeability is assessed. The permeability can be used to determine which cells are alive and which are dead. Dead cells are often gated out prior to gating debris and doublets. However, cleaning with respect to permeability is complicated in that it does not always make sense to clean "dead", or permeable, cells out of the data. Our default method chooses to classify them last because once an event is labeled "dead", it will not be assigned a different label. However, a user may wish to label the permeable, or dead cells right after cleaning the beads so that the "dead" label takes precedence over debris and doublets. Note that "dead" is used in place of permeable for labeling in this package.

Different event types are labeled iteratively so the `labels` vector in the `colData` will contain all of the labels and probabilities computed up to this point. Only events that have a "cell" label can be assigned an initial event classification of "dead". This function computes a score that assesses how much an event looks like a permeable cell and then fits a mixture model to assign each event a class of 1 for permeable, -1 for an event that is not permeable, or 0 for undetermined or previously assigned to a different event type. The score is recorded in the `score` object in the `colData` and the initial classification is recorded in the `initial` part of the `colData`.

The viability measures should classify into two fairly clear groups where one is permeable cells and the other is non-permeability. DNA is also often higher for cells that are permeable. The primary measure for determining permeability is sum of the viability measures, but the method allows for DNA content to be used as well. The function `initialGuess` is used to determine the groups. The members of the group with the largest mean are classified as 'dead' and the rest are classified as not dead.

**Value**

A `SingleCellExperiment` that contains the permeability score and the permeability designation for each event. This information is stored in the `score` and `initial` objects in the `colData` for the `SingleCellExperiment`.

**Examples**

```
data("raw_data", package = "CATALYST")
sce <- readCytobf(raw_data, beads = 'Beads', viability = c('cisPt1','cisPt2'))
sce <- initialBead(sce)
sce <- initialDebris(sce)
sce <- initialDoublet(sce)
sce <- initialDead(sce)
head(scores(sce))
head(initial(sce))
```

---

<code>initialDebris</code>	<i>Preliminary debris classification</i>
----------------------------	--

---

**Description**

Preliminary debris classification

**Usage**

```
initialDebris(x, score = c(1, 2, 3), standardize = TRUE)
```

**Arguments**

<code>x</code>	A <code>SingleCellExperiment</code> created with <code>readCytobf</code> .
<code>score</code>	A value of 1, 2, or 3 that specifies the debris score that should be calculated. See details for information on the debris score.
<code>standardize</code>	A value of <code>TRUE</code> will compute the debris score using standardized data. The raw data will be used to compute the score if <code>FALSE</code> . It is highly recommended that the data are standardized prior to computing the score because the variables are on different scales.

**Details**

The beads are typically the first cell classification that is done because their identification is straightforward. Debris is typically classified after the beads. This is because classifying debris is more straightforward than doublets and labeling them before the doublets aids in doublet classification.

Different event types are labeled iteratively so the `labels` vector in the `colData` will contain all of the labels and probabilities computed up to this point. Only events that have a "cell" label can be assigned an initial event classification of "debris". This function computes a score that assesses how much an event looks like debris and then fits a mixture model to assign each event a class of 1 for



debris, -1 for an event that is not debris, or 0 for undetermined or previously assigned to a different event type. The score is recorded in the `score` object in the `colData` and the initial classification is recorded in the `initial` part of the `colData`.

Several options are available for computing the debris score. The following list shows the debris score calculations. Each one can be selected by its number on the following list:

1.  $1 - (\text{DNA1} + \text{DNA2} + \text{Event\_length} - \text{Center} - \text{Width} + \text{Offset})$
2.  $\text{Residual} + \text{Offset} - 2(\text{DNA1}) - 2(\text{DNA2}) - \text{Event\_length} - \text{Center} - 0.5(\text{Width})$
3.  $1 - (\text{DNA1} + \text{DNA2} + \text{Event\_length})$

### Value

A `SingleCellExperiment` that contains the debris score and the debris designation for each event. This information is stored in the `score` and `initial` objects in the `colData` for the `SingleCellExperiment`.

### Examples

```
data("raw_data", package = "CATALYST")
sce <- readCytobf(raw_data, beads = 'Beads', viability = c('cisPt1','cisPt2'))
sce <- initialBead(sce)
sce <- initialDebris(sce)
head(scores(sce))
head(initial(sce))
```

---

<code>initialDoublet</code>	<i>Preliminary doublet classification</i>
-----------------------------	---

---

### Description

Preliminary doublet classification

### Usage

```
initialDoublet(x, score = c(1, 2, 3), standardize = TRUE)
```

### Arguments

<code>x</code>	A <code>SingleCellExperiment</code> created with <code>readCytobf</code> .
<code>score</code>	A value of 1, 2, or 3 that specifies the doublet score that should be calculated. See details for information on the doublet score.
<code>standardize</code>	A value of <code>TRUE</code> will use compute the doublet score using standardized data. The raw data will be used to compute the score if <code>FALSE</code> . It is highly recommended that the data are standardized prior to computing the score because the variables are on different scales.

## Details

The beads are typically the first cell classification that is done because their identification is straightforward. Debris is typically classified after the beads. This is because classifying debris is more straightforward than doublets and labeling them before the doublets aids in doublet classification.

Different event types are labeled iteratively so the `labels` vector in the `colData` will contain all of the labels and probabilities computed up to this point. Only events that have a "cell" label can be assigned an initial event classification of "doublet". This function computes a score that assesses how much an event looks like a doublet and then fits a mixture model to assign each event a class of 1 for doublet, -1 for an event that is not a doublet, or 0 for undetermined or previously assigned to a different event type. The score is recorded in the `score` object in the `colData` and the initial classification is recorded in the `initial` part of the `colData`.

Several options are available for computing the doublet score. The following list shows the doublet score calculations. Each one can be selected by its number on the following list:

1.  $\text{DNA1} + \text{DNA2} + \text{Residual} + \text{Event\_length} - \text{Offset} - 0.5(\text{Width})$
2.  $\text{DNA1} + \text{DNA2} + \text{Residual} + \text{Event\_length} - \text{Offset} - 0.5(\text{Width}) + \text{abs}(\text{Center})$
3.  $0.3 * (\text{DNA1} + \text{DNA2} + \text{Event\_length}) + \text{Residual} + \text{Center} + (\text{max}(\text{Offset}))$

## Value

A `SingleCellExperiment` that contains the doublet score and the doublet designation for each event. This information is stored in the `score` and `initial` objects in the `colData` for the `SingleCellExperiment`.

## Examples

```
data("raw_data", package = "CATALYST")
sce <- readCytobf(raw_data, beads = 'Beads', viability = c('cisPt1','cisPt2'))
sce <- initialBead(sce)
sce <- initialDebris(sce)
sce <- initialDoublet(sce)
head(scores(sce))
head(initial(sce))
```

---

`initialGuess`

*General preliminary classification.*

---

## Description

General preliminary classification.

## Usage

```
initialGuess(x, middleGroup = c(0, -1, 1))
```

**Arguments**

- `x` The score (ie. debris score, doublet score, etc.) to be used for predicting each event's label (eg. "doublet" vs. "cell").
- `middleGroup` numeric. When the optimal model (according to BIC) is the 3-component mixture model, this argument determines how to assign the middle group. Possible values are -1 for "cell", 0 (default) for "indeterminate", and 1 for the event type of interest (eg. "doublet").

**Value**

A list with the following elements:

- `label` A vector of the same length as `x` providing the labels (-1 for cells, 1 for non-cells, 0 for uncertain).
- `fit1` Summary of the 1-component (half Normal) model fit.
- `fit2` Summary of the 2-component (half Normal + Normal) model fit.
- `fit3` Summary of the 3-component (half Normal + 2 Normals) model fit.

**Examples**

```
data("raw_data", package = "CATALYST")
sce <- readCytobf(raw_data, beads = "Beads", viability = c("cisPt1", "cisPt2"))
sce <- initialDoublet(sce)
fit <- initialGuess(scores(sce, "doublet"))
```

---

labelQC

*Returns the final label assignments the specified parameters*

---

**Description**

Returns the final label assignments the specified parameters

**Usage**

```
labelQC(
  x,
  model = c("svm", "rf", "gbm"),
  type = c("all", "bead", "doublet", "debris", "dead"),
  nTrain = 4000,
  loss = c("auc", "class")
)
```

**Arguments**

x	A SingleCellExperiment created with <code>readCytof</code> with the scores and initial columns filled out for the event type of interest.
model	Type of model to use to do the labeling. Options are "svm" for a support vector machine, "gbm" for a gradient boosting machine, or "rf" for a random forest.
type	Types of events to model. Options are "all", "bead", "doublet", "debris", and "dead".
nTrain	The (maximum) number of data points to use when training a model to predict event types.
loss	Specifies the type of loss used to tune the GBM. Can be either "auc" for the area under the curve or "class" for classification error. This argument is ignored if random forest is used as the model.

**Details**

labelQC uses a support vector machine, gradient boosting machine, or a random forest to compute the final labels for the specified parameter types (bead, doublet, debris, or dead). The predicted probabilities for all of the observations are stored in the variable associated with that type for further analysis. Thus, it is possible to have a probability greater than 0.5 for 'debris' but still have a label of 'bead' if an observation was classified as a bead prior to classifying the debris.

**Value**

A SingleCellExperiment data.frame is returned with the labels for the parameters of listed in types (bead, doublet, debris, or dead) added to the label variable and the probabilities for each of the columns pertaining to the parameters listed in probs.

**Examples**

```
data("raw_data", package = "CATALYST")
sce <- readCytof(raw_data, beads = "Beads", viability = c("cisPt1", "cisPt2"))
sce <- labelQC(sce)
table(label(sce))
```

---

modelData	<i>Returns indices for data to be used to create the final classification model</i>
-----------	---

---

**Description**

Returns indices for data to be used to create the final classification model

**Usage**

```
modelData(x, type = c("bead", "doublet", "debris", "dead"), n = 4000)
```

**Arguments**

x	A SingleCellExperiment created with <a href="#">readCytof</a> with the scores and initial columns filled out for the event type of interest.
type	Identifies the type of label that is being modeled. Must be 'bead', 'doublet', 'debris', or 'dead'. Note that if no type of label is specified 'bead' will be used.
n	number of indices to return.

**Details**

The indices that are returned by `modelData` are be used to create a model that can be used to classify the observations with regard to the parameter of interest (bead, doublet, debris, dead). It is used as part of `gbmLabel`, `rfLabel`, `svmLabel`, and `labelQC`. The function `modelData` uses the score and the function `initialGuess` to randomly select a set of data points that we are confident are of the event type and not of the selected event type that can be used to train the data. Only points that are labeled as -1 and 1 are considered for the training dataset. The selected dataset is balance with a fairly equal number of points from each group.

**Value**

An integer vector that contains the indices of the events that should be included in the creation of the final classification model for the event type of interest (bead, debris, doublet, dead).

**Examples**

```
data("raw_data", package = "CATALYST")
sce <- readCytof(raw_data, beads = "Beads", viability = c("cisPt1", "cisPt2"))
sce <- initialBead(sce)
train <- modelData(sce, type = "bead", n = 4000)
```

---

plotInitialGuess      *Plot preliminary classification from initialGuess*

---

**Description**

Plot preliminary classification from initialGuess

**Usage**

```
plotInitialGuess(
  x,
  IG = NULL,
  fit = NULL,
  type = c("both", "full", "truncated")
)
```

**Arguments**

<code>x</code>	The score (ie. debris score, doublet score, etc.) to be used for predicting each event's label (eg. "doublet" vs. "cell").
<code>IG</code>	If <code>NULL</code> , the function <code>initialGuess</code> is used to fit a mixture of normal distributions. Otherwise, a numeric vector can be passed to the function that contains the fitted values.
<code>fit</code>	If left blank or <code>NULL</code> , the best fit as determined by BIC will be plotted. Otherwise, a numeric value of 1, 2, or 3 can be selected to plot single normal fit, mixture of two normals, or the mixture of three normals as fit by <code>initialGuess</code> .
<code>type</code>	Type of graph to be plotted. If 'truncated' is selected, only half of the first normal distribution will be plotted. If 'both' is selected, both the truncated and full plot will be plotted.

**Value**

A histogram that shows the score with the mixture of normal distributions overlaid.

**Examples**

```
data("raw_data", package = "CATALYST")
sce <- readCytof(raw_data, beads = "Beads", viability = c("cisPt1", "cisPt2"))
sce <- initialDoublet(sce)
plotInitialGuess(scores(sce, "doublet"), type = "both")
plotInitialGuess(scores(sce, "doublet"), type = "truncated")
```

---

readCytof

*Read in a dataset and prepare it for analysis*

---

**Description**

Read in a dataset and prepare it for analysis

**Usage**

```
readCytof(
  file.name,
  beads = c("Bead"),
  dna = c("DNA1", "DNA2"),
  event_length = "Event_length",
  viability = "Live_Dead",
  gaussian = c("Center", "Offset", "Width", "Residual"),
  verbose = TRUE
)
```

**Arguments**

file.name	A path to an .fcs file that contains CyTOF data or a flowSet object containing a single sample.
beads	character vector that contains the names of all of the bead channels.
dna	Character vector that contains the names of the DNA markers.
event_length	Character vector of the event length variable.
viability	Character vector of the permeability/viability markers.
gaussian	Character vector that contains the names of the Gaussian Discrimination Parameters.
verbose	Logical value indicating whether or not to print a summary of the technical channels identified in the data.

**Details**

The function returns a `SingleCellExperiment` that contains all of the original information from the fcs file. The data are imported using CATALYST and then information is added to the `colData` that will be used to determine labels for each event and to provide additional information about the events that can be used for exploratory data analysis and to aid the user in labeling the data. The objects are all initialized at this point and values are filled in during later stages of the labeling process. Note that the names from the fcs file are required as arguments to the `readCytof`. If you are not sure what those names are, there is some code in the example that shows how to import your data into a `SingleCellExperiment` using `prepData` from CATALYST and look at the names.

**Value**

A `SingleCellExperiment` that contains the information from the CyTOF fcs file, the technical data that will be used to label the data, and other objects that are used to store information through the labeling process. The objects are `DataFrame` objects that are stored in the `colData` for the `SingleCellExperiment`. The objects are:

label	A single variable <code>DataFrame</code> that will contain the event label as determined by <code>cytofQC</code> . At this point, all events are labeled "gdpZero" if <code>Event_length</code> or any of the Gaussian parameters are zero and "cell" otherwise. These labels are changed during later stages.
probs	A <code>DataFrame</code> that contains the "probability" that an event is a certain type. This is initialized as NA at this point and is filled in later on.
tech	A <code>DataFrame</code> that contains the technical variables used to determine the label of each event. The bead, DNA, and viability variables have an <code>arcsinh</code> transform, <code>Event_length</code> is unchanged, and the Gaussian parameters have a log transform using <code>log1p</code> .
scores	Scores are computed to determine how much an event looks like a bead, debris, doublet, or dead cell. These scores are used to select a training dataset for the classification model, but they can be helpful for exploratory data analysis so they are provided in this <code>DataFrame</code> . At this stage they are initialized as NA and values are added in later steps.

**initial** Initial classification of each event type is determined using a mixture model and the event type score. The `initial` object is a `DataFrame` that will hold this initial classification. A training dataset for the event classification model is selected using this initial classification.

### Examples

```
library(CATALYST)
library(SingleCellExperiment)
data("raw_data", package = "CATALYST")

# Determine at the names of the bead, DNA, and viability channels in the
# file. Names are 'Beads', 'DNA1', 'DNA2', 'cisPt1', 'cisPt2'.
tech <- prepData(raw_data)
rownames(tech)

# Determine names of event length and Gaussian parameters
# names are 'Event_length', 'Center', 'Offset', 'Width', 'Residual'
names(int_colData(tech))

# read in the data for use with cytofQC
x <- readCytotof(raw_data, beads = 'Beads', viability = c('cisPt1','cisPt2'))
```

---

rfLabel	<i>Returns the final label assignments for a parameter using a random forest</i>
---------	--

---

### Description

Returns the final label assignments for a parameter using a random forest

### Usage

```
rfLabel(
  x,
  type = c("bead", "doublet", "debris", "dead"),
  loss = c("auc", "class"),
  n = 4000,
  standardize = TRUE
)
```

### Arguments

x	A <code>SingleCellExperiment</code> created with <code>readCytotof</code> with the scores and initial columns filled out for the event type of interest.
type	Identifies the type of label that is being modeled. Must be 'bead', 'doublet', 'debris', or 'dead'.



loss	Specifies the type of loss used to tune the GBM. Can be either "auc" for the area under the curve or "class" for classification error.
n	number of observations in training dataset.
standardize	Indicates if the data should be standardized. Because the data are on different scales, it should be standardized for this analysis because the variables are on different scales.

## Details

rfLabel uses a random forest to compute the final labels for the specified parameter type (bead, doublet, debris, or dead). This step cannot be completed until the corresponding initialization function (initialBead, initialDebris, initialDoublet, or initialDead) is done on the SingleCellExperiment created by readCytof. The random forest uses the defaults from randomForest and then predicted values are computed for all of the events in x. If the predicted probability for the label type is greater than 0.5, the label is changed to the specified type. However, if an observation already has a label other than 'cell' in the label variable, it will not be changed. The predicted probabilities for all of the observations are stored in the variable associated with that type in the probs object of x for further analysis. Thus, it is possible to have a probability greater than 0.5 for 'debris' but still have a label of 'bead' if an observation was classified as a bead prior to classifying the debris.

## Value

An updated SingleCellExperiment is returned with the labels for the parameter of interest (bead, doublet, debris, or dead) added to the label object of the SingleCellExperiment and the probabilities for the event type added to the probs object of the SingleCellExperiment.

## Examples

```
data("raw_data", package = "CATALYST")
sce <- readCytof(raw_data, beads = "Beads", viability = c("cisPt1", "cisPt2"))
sce <- initialBead(sce)
sce <- rfLabel(sce, type = "bead")
head(probs(sce))
table(label(sce))
```

---

s3vmLabel

*Returns the final label assignments for a parameter using a semi-supervised support vector machine*

---

## Description

Returns the final label assignments for a parameter using a semi-supervised support vector machine

**Usage**

```
s3vmLabel(
  x,
  type = c("bead", "doublet", "debris", "dead"),
  loss = c("auc", "class"),
  n = 4000,
  standardize = TRUE
)
```

**Arguments**

x	A SingleCellExperiment created with <a href="#">readCytobf</a> with the scores and initial columns filled out for the event type of interest.
type	Identifies the type of label that is being modeled. Must be 'bead', 'doublet', 'debris', or 'dead'.
loss	Specifies the type of loss used to tune the SVM. Can be either "auc" for the area under the curve or "class" for classification error.
n	number of observations in training dataset.
standardize	Indicates if the data should be standardized. Because the data are on different scales, it should be standardized for this analysis because the variables are on different scales.

**Details**

s3vmLabel uses a semi-supervised support vector machine to compute the final labels for the specified parameter type (bead, doublet, debris, or dead). The model is initially computed using only the data specified in the index argument. Events are iteratively added to this set when the updated SVM predicts a label with high confidence. Then predicted values are computed for all of the observations in x. If the predicted probability for the label type is greater than 0.5, the label is changed to the specified type. However, if an observation already has a label other than 'cell' in the labels\$label variable, it will not be changed. The predicted probabilities for all of the observations is stored in the variable associated with that type for further analysis. Thus, it is possible to have a probability greater than 0.5 for 'debris' but still have a label of 'bead' if an observation was classified as a bead prior to classifying the debris.

**Value**

An updated SingleCellExperiment is returned with the labels for the parameter of interest (bead, doublet, debris, or dead) added to the label object of the SingleCellExperiment and the probabilities for the event type added to the probs object of the SingleCellExperiment.

**Examples**

```
data("raw_data", package = "CATALYST")
sce <- readCytobf(raw_data, beads = "Beads", viability = c("cisPt1", "cisPt2"))
sce <- initialBead(sce)
sce <- svmLabel(sce, type = "bead", loss = "auc")
head(probs(sce))
```

```
table(label(sce))
```

---

scores

*Returns a specified object from the cytofQC SingleCellExperiment*

---

### Description

Returns a specified object from the cytofQC SingleCellExperiment

### Usage

```
scores(x, type = c("all", "bead", "debris", "doublet", "dead"))
probs(x, type = c("all", "bead", "debris", "doublet", "dead"))
label(x)
tech(
  x,
  type = c("all", "Bead", "DNA", "Viability", "Event_length", "Center", "Offset",
    "Width", "Residual")
)
initial(x, type = c("all", "bead", "debris", "doublet", "dead"))
```

### Arguments

x	A SingleCellExperiment created with <a href="#">readCytobf</a> with the scores and initial columns filled out for the event type of interest.
type	Identifies the type of objects to be returned. For scores and probs, type can be one or more of 'bead', 'dead', 'debris', or 'doublet'. For tech it can be any of the QC variables. It will return the numeric vector or DataFrame for the score(s) or probability for the specified event type(s). If the event types are not specified, the DataFrame containing all of the scores or all of the probability will be returned. This argument does nothing for label.

### Value

For probs, scores, and tech, a numeric vector or DataFrame with the information for the event type(s) is returned.

For label, a character vector containing the label for each event is returned.

For tech, a DataFrame containing the technical variables used to determine the label of each event. The bead, DNA, and viability variables have an arcsinh transform, Event\_length is unchanged, and the Gaussian parameters have a log transform using log1p.

## Examples

```
data("raw_data", package = "CATALYST")
sce <- readCytof(raw_data, beads = "Beads", viability = c("cisPt1", "cisPt2"))
sce <- labelQC(sce)
table(label(sce))
cytofHist(scores(sce, type = 'bead'), label(sce), title = "Bead score")
```

---

svmLabel	<i>Returns the final label assignments for a parameter using a support vector machine</i>
----------	---

---

## Description

Returns the final label assignments for a parameter using a support vector machine

## Usage

```
svmLabel(
  x,
  type = c("bead", "doublet", "debris", "dead"),
  loss = c("auc", "class"),
  n = 4000,
  standardize = TRUE
)
```

## Arguments

x	A SingleCellExperiment created with <a href="#">readCytof</a> with the scores and initial columns filled out for the event type of interest.
type	Identifies the type of label that is being modeled. Must be 'bead', 'doublet', 'debris', or 'dead'.
loss	Specifies the type of loss used to tune the GBM. Can be either "auc" for the area under the curve or "class" for classification error.
n	number of observations in training dataset.
standardize	Indicates if the data should be standardized. Because the data are on different scales, it should be standardized for this analysis because the variables are on different scales.

## Details

svmLabel uses a support vector machine to compute the final labels for the specified parameter type (bead, doublet, debris, or dead). This step cannot be completed until the corresponding initialization function (`initialBead`, `initialDebris`, `initialDoublet`, or `initialDead`) is done on the SingleCellExperiment created by `readCytof`. The support vector machine is tuned using [eztune](#) and then predicted values are computed for all of the events in x. If the predicted probability for the label type is greater than 0.5, the label is changed to the specified type. However, if an observation

already has a label other than 'cell' in the label variable, it will not be changed. The predicted probabilities for all of the observations are stored in the variable associated with that type in the probs object of x for further analysis. Thus, it is possible to have a probability greater than 0.5 for 'debris' but still have a label of 'bead' if an observation was classified as a bead prior to classifying the debris.

### Value

An updated SingleCellExperiment is returned with the labels for the parameter of interest (bead, doublet, debris, or dead) added to the label object of the SingleCellExperiment and the probabilities for the event type added to the probs object of the SingleCellExperiment.

### Examples

```
data("raw_data", package = "CATALYST")
sce <- readCytof(raw_data, beads = "Beads", viability = c("cisPt1", "cisPt2"))
sce <- initialBead(sce)
sce <- svmLabel(sce, type = "bead", loss = "auc")
table(label(sce))
```

# Index

cytofHist, [2](#), [3](#)  
cytofQC (cytofQC-package), [2](#)  
cytofQC-package, [2](#)  
cytofQCreport, [2](#), [4](#)

eztune, [5](#), [20](#)

gbmLabel, [4](#)

initial (scores), [19](#)  
initialBead, [6](#)  
initialDead, [7](#)  
initialDebris, [8](#)  
initialDoublet, [9](#)  
initialGuess, [10](#)

label (scores), [19](#)  
labelQC, [2](#), [4](#), [11](#)

modelData, [12](#)

plotInitialGuess, [13](#)  
probs (scores), [19](#)

readCytof, [2](#), [5–9](#), [12](#), [13](#), [14](#), [16](#), [18–20](#)  
rfLabel, [16](#)

s3vmLabel, [17](#)  
scores, [19](#)  
SingleCellExperiment, [2](#)  
svmLabel, [20](#)

tech (scores), [19](#)