Package 'flowSpecs'

November 12, 2026

Version 1.26.0

Date 2026-03-16

Type Package

Title Tools for processing of high-dimensional cytometry data

biocViews Software,CellBasedAssays,DataRepresentation,ImmunoOncology, FlowCytometry,SingleCell,Visualization,Normalization,DataImport

Description This package is intended to fill the role of conventional cytometry pre-processing software, for spectral decomposition, transformation, visualization and cleanup, and to aid further downstream analyses, such as with DepecheR, by enabling transformation of flowFrames and flowSets to dataframes. Functions for flowCore-compliant automatic 1D-gating/filtering are in the pipe line.
The package name has been chosen both as it will deal with spectral cytometry and as it will hopefully give the user a nice pair of spectacles through which to view their data.

BugReports https://github.com/jtheorell/flowSpecs/issues

License MIT + file LICENSE

Encoding UTF-8

LazyData false

RoxygenNote 7.3.2

Depends R (>= 4.3)

Imports ggplot2 (>= 3.1.0), BiocGenerics (>= 0.30.0), BiocParallel (>= 1.18.1), Biobase (>= 2.48.0), reshape2 (>= 1.4.3), flowCore (>= 1.50.0), zoo (>= 1.8.6), stats (>= 3.6.0), methods (>= 3.6.0)

Suggests testthat, knitr, rmarkdown, BiocStyle, DepecheR

VignetteBuilder knitr

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

git_branch RELEASE_3_23

git_last_commit 03b6eeb

git_last_commit_date 2026-04-28

Repository Bioconductor 3.23

Date/Publication 2026-11-11

Author Jakob Theorell [aut, cre]

Maintainer Jakob Theorell <jakob.theorell@ki.se>

Contents

flowSpecs-package 2
arcTrans 3
correctUnmix 4
corrMatCreate 5
flowSet2LongDf 6
fullPanel 7
newExprs<- 8
oneVsAllPlot 8
peakIdenti 10
peakNorm 11
specMat 12
specMatCalc 13
specUnmix 14
unmixCtrls 15

Index **16**

`flowSpecs-package` *Tools for processing high-dimensional cytometry files*

Description

This package is intended to fill the role of conventional cytometry pre-processing software, for spectral decomposition, transformation, visualization and cleanup, and to aid further downstream analyses, such as with DepecheR, by enabling transformation of flowFrames and flowSets to dataframes. Functions for flowCore-compliant automatic 1D-gating/filtering are in the pipe line. It is worth noting here that even if there are dedicated spectral cytometers, it is possible to increase the separation of the fluorochromes in a conventional flow cytometer too, by just keeping all non-used channels open. That will however also require the use of spectral unmixing, rather than compensation, as the compensation functions generally require the compensation matrix to be symmetrical. So please open all channels, and use this software!

Author(s)

Maintainer: Jakob Theorell <jakob.theorell@ki.se>

See Also

`flowCore`

arcTrans *Efficient inverse hyperbolic cosine transformation*

Description

This is a simple wrapper function for the base asinh function, that is useful for flowFrames and flowSets. It also allows for reversing the transformation with the argument "unTrans".

Usage

```
arcTrans(flowObj, transNames, transCoFacs = "default", unTrans = FALSE)
```

Arguments

flowObj	The fcs object to be transformed. Both flowFrames and flowSets are accepted.
transNames	The variables that should be normalized.
transCoFacs	This value or vector of values define the values for the transformation during the normalization. In the "default" case, the function defines the object as a CyTOF object if >5 percent of the values are 0, and applies the transformation value 5. Otherwise, the value 400 is applied.
unTrans	If the reverse action should be taken, i.e. if an already transformed dataset should be un-transformed. NB! It is of great importance that the same transformation factors are used!

Value

A flow object containing the transformed data, and with all metadata left untouched.

Examples

```
# Import some data and the spectral matrix. The latter can be generated using
# specMatCalc
data(fullPanel)
data(specMat)

fullPanelUnmixed <- specUnmix(fullPanel, specMat)

# Identify the columns that should be transformed
colnames(fullPanelUnmixed)
# The time and scatter parameters should not, but apart from that, all should
# be included.
transNames <- colnames(fullPanelUnmixed)[seq(6,18)]

# ow, transform this file, with the default transformation factor of 400.
# NB! It is alway advisable to visually (or computationally) check the data
# for the most optimal transformation factors. These often vary from marker
# to marker.
```

```
fullPanelTrans <- arcTrans(fullPanelUnmixed, transNames)
```

`correctUnmix` *Correct defects in spectral unmixing by compensation*

Description

This function provides a way to reduce the defects in the spectral unmixing, by creating a secondary correction matrix, which is symmetrical.

Usage

```
correctUnmix(unmixFlowObj, corrMat, transCoFacs = 400)
```

Arguments

`unmixFlowObj`	A flowframe or flowset post unmixing.
`corrMat`	A correction matrix. If this is the first round, the executionof this function needs to be preceeded by the generation of this matrix, for example by using the `corrMatCreate` function.
`transCoFacs`	If transformation should be performed, the transformation cofactors can be added here. Three possible inputs: a vector with specific cofactors for each variable, a set value that will be used for all variables, and FALSE. Note: It might be good to set this to FALSE in the final round, to optimize the transoformations externally.

Value

The unmixed flow object, now corrected with the values from the corrMat.

See Also

`specUnmix`, `arcTrans`, `corrMatCreate`

Examples

```
# Load uncompensated data
data(fullPanel)

# Load the spectral unmixing matrix generated with controls from the same
# experiment. These can be generated using the specMatCalc function.
data(specMat)

# And now unmix
fullPanelUnmix <- specUnmix(fullPanel, specMat)
```

```
# Create an empty unmixinng matrix
corrMat <- corrMatCreate(specMat)

# Now correct the data with this. In the first instance, this will of course
# not have any effect, more  than transformation, as the corrMat is empty.
fullPanelCorr <- correctUnmix(fullPanelUnmix, corrMat)

# This now needs to be investigated, to identify any possible compensation
# defects. This is most easily done with the oneVsAllPlot executed in the
# following way:
## Not run:
oneVsAllPlot(fullPanelCorr)

## End(Not run)
# One obvoius defect that shows when doing this is between CD56 and IgM:
oneVsAllPlot(fullPanelCorr, "BV650_CD56", saveResult = FALSE)

# This is corrcted the following way:
corrMat["BV650_CD56", "AF647_IgM"] <- -0.03
fullPanelCorr <- correctUnmix(fullPanelUnmix, corrMat)
oneVsAllPlot(fullPanelCorr, "BV650_CD56", saveResult = FALSE)

# This process is iterated until there are no remaining artifacts. Good help
# to do this is a set of fluorescence-minus-one controls. If that is not
# available, a rule of thumb is that if the signal in marker x is
# strongly negatively correlated to marker y, so that highly
# single-x-posisive values are below zero, then this is with all likelihood
# an artifact. The situation becomes more complicated with strong positive
# correlations, as they can occur in biology, so there one has to take more
# care and keep the marker biology in mind.
```

`corrMatCreate` *Generate a correction matrix for cytometry data analysis*

Description

This function aids the correctUnmix function, to create a symmetrical correction matrix that should be used together with a flowframe to correct the errors of unmixing.

Usage

```
corrMatCreate(specMat)
```

Arguments

`specMat`	The spectral matrix used to unmix the dataset of interest.

Value

A symmetrical matrix of zeros with the right row- and column names.

See Also

correctUnmix

Examples

```
# Load uncompensated data
data(fullPanel)

# Load the spectral unmixing matrix generated with controls from the same
# experiment. These can be generated using the specMatCalc function.
data(specMat)

# And now unmix
fullPanelUnmix <- specUnmix(fullPanel, specMat)


# Create an empty unmixinng matrix
corrMat <- corrMatCreate(specMat)
```

`flowSet2LongDf` *Convert a flowSet to one long dataframe with all identifiers as separate #columns.*

Description

This function is mainly used for compatibility with matrix-based clustering algorithms, such as depeche in the DepecheR package.

Usage

```
flowSet2LongDf(flowObj, idInfo)
```

Arguments

`flowObj`	The flowSet or flowFrame to be converted to a dataframe.
`idInfo`	A list of any number of characteristics that can be derived from the file names. For each entry, a gsub specification of where to find the information in the file name should be added, such as id=""..._|..."".

Value

A long data frame with one column per PMT/APD (or fluorochrome, depending on the state of the imported files), one for the acquisition date (for fcs files) and one colum for each specified slot above. If no gsub-pattern is provided, only a single column with the full file name will be used to separate the observations from each file.

See Also

depeche

Examples

```
#' # Load uncompensated data
data(fullPanel)

# Load the spectral unmixing matrix generated with controls from the same
# experiment. These can be generated using the specMatCalc function.
data(specMat)

# Now unmix
fullPanelUnmix <- specUnmix(fullPanel, specMat)

# Transform all fluorescent channels
fullPanelTrans <- arcTrans(fullPanelUnmix,
    transNames = colnames(fullPanelUnmix)[6:18])

# This function is primarily meant to be used with flowSets.
# If we had only one flowFrame, we could just extract the data by
# the use of the flowCore function exprs(), so we will convert the data to a
# flowSet now.
library(flowCore)
fullPanelFs <- flowSet(fullPanelTrans)

# Before converting to a dataframe it is important to get an idea of the
# structure of the names, to be able to extract meaningful parts of the name.
# Here, we have an exceptional case again, as the flowSet has just been
# created, so there is actually no meaningful name of the flowFrame inside
# it. So for example reasons, we will give it one now:
sampleNames(fullPanelFs) <- "PBMC_full_panel_d1.fcs"

# And now, we generate the dataframe:
fullPanelDf <- flowSet2LongDf(fullPanelFs, idInfo =
                                  list("Tissue" = "|_full_panel_..\\.fcs",
                                      "Donor" = "...._full_panel_|\\.fcs"))
# This is the result
str(fullPanelDf)
```

`fullPanel` *A fully stained spectral cytometry sample*

Description

This is a flowFrame with a PBMC sample stained with 12 fluorochromes. Data acquired on a 44 detector, 3 laser Cytek Aurora® instrument by J Theorell. Date: 2018-10-25.

Usage

```
data(fullPanel)
```

Format

An object of class "flowFrame"

`newExprs<-` *This function lets us exchange a flowframes exprs portion to an unrelated one It is solely meant to be used internally, as it is a strange practice.*

Description

This function lets us exchange a flowframes exprs portion to an unrelated one It is solely meant to be used internally, as it is a strange practice.

Usage

```
newExprs(x) <- value
```

Arguments

x	A flowFrame
value	A matrix suitable to be an exprs object.

Value

A new flowFrame with "value" as the exprs portion.

`oneVsAllPlot` *Plotting all variables against a single variable*

Description

This function is useful both when setting appropriate gates and when the adjustments of the compensation are done

Usage

```
oneVsAllPlot(
  flowObj,
  yCol = "all",
  nRows = 10000,
  plotName = "default",
  dirName = "All_vs_all_plots",
  zeroTrim = TRUE,
  hexBins = 30,
  saveResult = TRUE
)
```

Arguments

flowObj	This is the full dataset, either a flowFrame or a flowSet, that should be plotted. If it has more rows than "nRows", a subsample (with equal contributions from each sample if a flowSet) will be plotted.
yCol	Here, the variable to be plotted against all the others is selected. It can be either a number, the column name of interest or "all".
nRows	The number of rows that will be used to construct the plot. The fewer, the faster, but the resolution also decreases, naturally. Default is 100000.
plotName	If a name different from yCol should be used, it can be added here.
dirName	Add a custom directory name.
zeroTrim	In the case of CyTOF data, the events at zero can often be so dominant, that all other density variation is dwarfed, and thus invisible. With this command, the events that are zero in both y and x are trimmed for each x separately.
hexBins	How many bins should the hexagonal plots be divided in?
saveResult	Should the result be saved as a file?

Value

A plot with one 2D-graph for each variable that the y-variable should be plotted against.

See Also

correctUnmix

Examples

```
#' # Load uncompensated data
data(fullPanel)

# Load the spectral unmixing matrix generated with controls from the same
# experiment. These can be generated using the specMatCalc function.
data(specMat)

# Now unmix
```

```
fullPanelUnmix <- specUnmix(fullPanel, specMat)

# Transform all fluorescent channels
fullPanelTrans <- arcTrans(fullPanelUnmix,
    transNames = colnames(fullPanelUnmix)[6:18])

# And now run the function. If no specific marker is selected, as in this
# case, then all markers will be plotted in a new sub-directory.
# Further, if you leave the saveResult to TRUE, a pdf will be created.
oneVsAllPlot(fullPanelTrans, yCol = "BV650_CD56", saveResult = FALSE)

# This shows that there is a compensation artifact between AF647_IgM and
# BV650_CD56, which is an expected combination to cause problems, due to the
# similar emission characteristics. It is therefore recommended to go on to
# correctUnmix function.
```

`peakIdenti` *Peak identification for higher-level functions.*

Description

This function is primarily thought to be used internally to define peaks in data. One function is borrowed from package vulcan, namely the vulcan::densityauc, which is neat, but the package is large and significantly increases the installation time, and the importing is thus discarded.

Usage

```
peakIdenti(
  markerData,
  volThresh = 0.05,
  distThresh = 0.1,
  adjust = 2,
  nPeaks = 2,
  returnStats = FALSE
)
```

Arguments

`markerData`	The data that the peaks should be identified for
`volThresh`	The cutoff ratio of the volume for each secondary peak, under which it is not considered to be a peak
`distThresh`	The cutoff under which two peaks are considered one, as they are too close to each other. This value between 0 and 1 corresponds to a fraction from the 10th to the 90th percentile of the data range that the peaks must be separated by to count. Defaults to 0.1 or 10 percent of the distance.
`adjust`	The value deciding the accuracy of the density calculation. The higher the value, the lower the sensitivity for small aberrations in the density.

nPeaks	The number of peaks that should be exported. If n+1 fulfilling the volRatio criterion are found, the peaks most separated in space are chosen.
returnStats	Should the deflection points defining the peaks, the peak hight and the lowest deflection point between the two most extreme peaks be included in the export?

Value

The information about the peaks in question. Depending on if returnStats is TRUE or not and the number of peaks, it will change in complexity.

See Also

densityauc

---

peakNorm *Normalize batch differences in intensities by aligning peaks*

Description

This function is intended to be used on standardized controls, preferrably one standard that has been acquired with every batch. This function needs to be applied separately for each batch, with the same standard.

Usage

```
peakNorm(
  fs,
  ctrlPos,
  standFF,
  transNames = FALSE,
  transCoFacs,
  volThresh = 0.005,
  normOrNot = rep(TRUE, ncol(standFF))
)
```

Arguments

fs	The flowSet to be normalized
ctrlPos	The position in the flowSet that contains the internal control
standFF	The external standard to which all batches should be normalized
transNames	If parameters are not transformed prior to running this, internal transformation is necessary to identify peaks, so specify which variables that need transformation here. The data for these variables are untransformed at the end, so the data will have the same scale in and out.
transCoFacs	Also inherited from arcTrans. Low values (2-10) for CyTOF data, high values (200-2000) for flow data, all depending on the number of input channels.

volThresh	The threshold for how small the area of a peak can be compared to the largest peak, and still count.
normOrNot	This needs to be a logical vector with the same length as the number of columns in the fs and standFF. If it is known that a certain variable should not be normalized, that information can be specified here.

Value

A normalized flowSet. In addition, output to console, to clarify #for which markers the same number of peaks was detected in the control and the standard, as normalization will only be applied for these.

Examples

```
#Load uncompensated data and spectral matrix.
data(fullPanel)
data(specMat)

# And now unmix
fullPanelUnmix <- specUnmix(fullPanel, specMat)

 #Create a new file with the value 1000 added to all values
 library(flowCore)
 fullPanelPlus1000 <- flowFrame(exprs(fullPanelUnmix)+1000)
# Check how they differ.
range(exprs(fullPanelUnmix)[,1])
# 143 1187733
range(exprs(fullPanelPlus1000)[,1])
# 1143 1188733

#Now normalize the new one to the old. NB! Here we will only
normPanel1000 <- peakNorm(flowSet(fullPanelPlus1000), 1, fullPanelUnmix,
transNames = colnames(fullPanelUnmix)[6:18], transCoFacs = 500)

#And now check the new result
range(exprs(normPanel1000[[1]] )[,1])
# 143 1187733
```

---

specMat *Spectral unmixing matrix*

Description

This matrix is generated using the specMatCalc function and the unmixCtrls example file.

Usage

```
data(specMat)
```

Format

An object of class "matrix";

---

specMatCalc *Calculating the matrix used for spectral unmixing*

Description

This algoritm takes a flowSet containing single-stained controls and negative controls, including an autofluorescence control and estimates the unmixing for all fluorescent variables.

Usage

```
specMatCalc(
  unmixCtrls,
  groupNames,
  autoFluoName,
  scatterGating = TRUE,
  excludedColumnNamePatterns = "Time|SC|Sort|ate|omp|LightLoss|Row|Column"
)
```

Arguments

unmixCtrls	A flowSet containing all the single stained and unstained files necessary to create an spectral unmixing matrix. These can but do not have to, contain a negative control. Such a negative control will not be used, and instead an universal negative control needs to be included for each sample type present among the single-stained controls.
groupNames	A character vector containing strings common to the groups of non-autofluoresence unmixCtrls that could be present. If for example all antibodies single stains are anti-mouse bead-based the dead cell marker is stained PBMC, and the files congruently either have a prefix containing "Bead" or "PBMC", then the vector should be c("Bead", "PBMC"). The system is not case specific.
autoFluoName	The sample name of the autofluorescence control.
scatterGating	Should an internal gate be created based on the highest peak in forward- and side scatter? Defaults to TRUE for backward compatibility.
excludedColumnNamePatterns	Here, the columns that do not have spectral information are identified and excluded.

Value

A data frame with each row representing a fluorochrome or or autofluorescence and each column representing a detector.

Examples

```
# Load suitable unmixing controls. NB! If these originate from different
# sample types, such as beads and PBMC, there should be a negative control
# for each group and the names should reflect this, so that all PBMC samples
# would be called PBMC_unstained, PBMC_DCM, etc.
data(unmixCtrls)

# If the dataset contains cell controls, make sure that the cell population
# interest dominates FSC-A, as the data highest peak in this channel will be
# used.

# And run the function
specMat <- specMatCalc(unmixCtrls, groupNames = c("Beads_", "Dead_"),
autoFluoName = "PBMC_unstained.fcs")
```

---

specUnmix *Spectral unmixing of cytometry files*

Description

This function performs the central task of spectral unmixing, to convert the raw photon detector input to "biological" proxy-signals.

Usage

```
specUnmix(flowObj, specMat)
```

Arguments

flowObj	The fcs object to be filtered. Both flowFrames and flowSets are accepted.
specMat	This is a matrix generated by the secMatCalc function, possibly with edited row names.

Value

The unmixed data. It will be returned in the format it was imported as.

Examples

```
# Load uncompensated data
data(fullPanel)

# Load the spectral unmixing matrix generated with controls from the same
# experiment. These can be generated using the specMatCalc function.
data(specMat)

# And now, just run the function
fullPanelUnmix <- specUnmix(fullPanel, specMat)
```

unmixCtrls *Unmixing controls*

Description

This is a flowSet with 14 spectral unmixing controls: 11 single-stained bead populations, 1 unstained bead, one dead-cell-marker- stained PBMC sample, one unstained PBMC sample, working as a control for the dead cell marker, and one autofluorescence control, which is also unstained PBMC (in fact the same sample as the negative control for the dead cell marker). Data acquired on a 44 detector, 3 laser Cytek Aurora® instrument by J Theorell. Date: 2018-10-25.

Usage

```
data(unmixCtrls)
```

Format

An object of class `"flowSet"`

Index

∗ **datasets**
    fullPanel, 7
    specMat, 12
    unmixCtrls, 15
∗ **internal**
    newExprs<-, 8
    peakIdenti, 10
∗ **package**
    flowSpecs-package, 2

arcTrans, 3, *4*

correctUnmix, 4, *6*, *9*
corrMatCreate, *4*, 5

densityauc, *11*
depeche, *7*

flowCore, *2*
flowSet2LongDf, 6
flowSpecs *(*flowSpecs-package*)*, 2
flowSpecs-package, 2
fullPanel, 7

newExprs<-, 8

oneVsAllPlot, 8

peakIdenti, 10
peakNorm, 11

specMat, 12
specMatCalc, 13
specUnmix, *4*, 14

unmixCtrls, 15