Package ‘flowSpecs’

October 17, 2020

Version 1.2.0

Date 2020-04-27

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 7.1.0

Depends R (>= 3.6)

Imports ggplot2 (>= 3.1.0), BiocGenerics (>= 0.30.0), BiocParallel (>=
1.18.1), 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, flowVS, DepecheR

VignetteBuilder knitr

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

git_branch RELEASE_3_11

git_last_commit 5399350

git_last_commit_date 2020-04-27

Date/Publication 2020-10-16

Author Jakob Theorell [aut, cre]

Maintainer Jakob Theorell <jakob.theorell@ndcn.ox.ac.uk>

R topics documented:

`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@ndcn.ox.ac.uk>

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 check the data for the most optimal
# transformation factors. For flow cytometry data, this can potentially be
# done using the flowVS package:
# https://www.bioconductor.org/packages/release/bioc/html/flowVS.html

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"

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",
  zeroTrim = TRUE,
  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.
`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.
`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.
```

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)
```

Arguments

`unmixCtrls`	A flowSet containing all the single stained and unstained files necessary to create an spectral unmixing matrix.
`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.

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