Package ‘cytometree’

December 4, 2019

Type Package

Title Automated Cytometry Gating and Annotation

Version 2.0.2

Date 2019-12-04

Author Chariff Alkhassim, Boris P Hejblum, Anthony Devaux, Van Hung Huynh Tran

Maintainer Boris P Hejblum <`boris.hejblum@u-bordeaux.fr`>

LinkingTo Rcpp, RcppArmadillo

Description Given the hypothesis of a bi-modal distribution of cells for each marker, the algorithm constructs a binary tree, the nodes of which are subpopulations of cells. At each node, observed cells and markers are modeled by both a family of normal distributions and a family of bi-modal normal mixture distributions. Splitting is done according to a normalized difference of AIC between the two families. Method is detailed in: Commenges, Alkhassim, Gottardo, Hejblum & Thiebaut (2018) <doi: 10.1002/cyto.a.23601>.

License LGPL-3 | file LICENSE

LazyData true

Depends R (>= 3.1.0), Rcpp (>= 0.12.11)

Imports ggplot2, graphics, igraph, mclust, methods, stats, cowplot, GoFKernel

RoxygenNote 7.0.2

BugReports `https://github.com/sistm/Cytometree/issues`

Encoding UTF-8

Suggests knitr, formatR, rmarkdown, viridis

VignetteBuilder knitr

NeedsCompilation yes

Repository CRAN

Date/Publication 2019-12-04 22:20:19 UTC

R topics documented:

`cytometree-package` *Binary tree algorithm for cytometry data analysis.*

Description

The algorithm is based on the construction of a binary tree, the nodes of which are subpopulations of cells. At each node, observed cells and markers are modeled by both a family of normal distributions and a family of bi-modal normal mixture distributions. Splitting is done according to a normalized difference of AIC between the two families. Given the unsupervised nature of the binary tree, some of the available markers may not be used to find the different cell populations present in a given sample. To recover a complete annotation, we defined, as a post processing procedure, an annotation method which allows the user to distinguish two or three expression levels per marker.

Details

Package:	cytometree
Type:	Package
Version:	2.0.2
Date:	2019-12-04
License:	LGPL-3

The main function in this package is `CytomeTree`.

Author(s)

Chariff Alkhassim, Boris Hejblum, Anthony Devaux, Van Hung Huynh Tran — Maintainer: Boris Hejblum

References

Commenges D, Alkhassim C, Gottardo R, Hejblum BP, Thiébaut R (2018). cytometree: a binary tree algorithm for automatic gating in cytometry analysis. Cytometry Part A, 93(11):1132-1140. <doi: 10.1002/cyto.a.23601>

`Annotation` *Annotates cell populations found using CytomeTree.*

Description

Annotates cell populations found using CytomeTree.

Usage

```
Annotation(
  CytomeTreeObj,
  K2markers = NULL,
  K3markers = NULL,
  plot = TRUE,
  t = 0.2,
  remove_outliers_inplot = TRUE,
  center_fun = c("median", "mean")
)
```

Arguments

`CytomeTreeObj`	An object of class CytomeTree.
`K2markers`	A vector of class character where the names of the markers for which 2 levels of expression are sought can be specified. Default is `NULL` i.e. unsupervised.
`K3markers`	A vector of class character where the names of the markers for which 3 levels of expression are sought can be specified. Default is `NULL` i.e. unsupervised.
`plot`	A logical value indicating whether or not to plot the partitioning in 1, 2 or 3 groups for each marker. Default is `TRUE`.
`t`	A real positive-or-null number used for comparison with the normalized AIC computed to compare the fits of the marginal distributions obtained by one normal distribution and by a mixture of two or three normal. For markers used in the tree, the algorithm compares the fits obtained by a mixture of two and three normal distributions. Default value is .2. A higher value leads to a smaller number of expression levels per marker.
`remove_outliers_inplot`	a logical flag indicating whether the y-axis should be scaled by removing outliers or not. Default is `TRUE`.
`center_fun`	a character string either 'median' or 'mean' indicating based on which summary the populations should be ordered. Default is `'median'`, which is more robust to outliers and long tail distributions.

Details

The algorithm is set to find the partitioning in 1, 2 or 3 groups of cell populations found using CytomeTree. In an unsupervised mode, it minimizes the within-leaves sum of squares of the observed values on each marker and computes the normalized AIC to compare the fits of the marginal distributions obtained by one normal distribution and by a mixture of two or three normal.For markers used in the tree, the algorithm compares the fits obtained by a mixture of two and three normal distributions.

Value

A `data.frame` containing the annotation of each cell population.

Author(s)

Chariff Alkhassim, Boris Hejblum

`bootstrapCI` *Bootstrapped Confidence Interval.*

Description

Bootstrapped Confidence Interval.

Usage

```
bootstrapCI(stat, n, alpha)
```

Arguments

`stat`	A numeric vector of statistics for which to compute a confidence interval.
`n`	An integer giving the number of bootstrap samples.
`alpha`	A real number comprised in]0, 1[: 1 - desired confidence level.

Author(s)

Chariff Alkhassim

`CytofTree` *Binary tree algorithm for mass cytometry data analysis.*

Description

Binary tree algorithm for mass cytometry data analysis.

Usage

```
CytofTree(
  M,
  minleaf = 1,
  t = 0.1,
  verbose = TRUE,
  force_first_markers = NULL,
  transformation = c("asinh", "biexp", "log10", "none"),
  num_col = 1:ncol(M)
)
```

Arguments

`M`	A matrix of size n x p containing mass cytometry measures of n cells on p markers.
`minleaf`	An integer indicating the minimum number of cells per population. Default is `1`.
`t`	A real positive-or-null number used for comparison with the normalized AIC computed at each node of the tree. A higher value limits the height of the tree.
`verbose`	A logical controlling if a text progress bar is displayed during the execution of the algorithm. By default is TRUE.
`force_first_markers`	a vector of index to split the data on first. This argument is used in the semi-supervised setting, forcing the algorithm to consider those markers first, in the order they appear in this `force_first_markers` vector, and forcing the split at every node. Default is `NULL`, in which case the clustering algorithm is unsupervised.
`transformation`	A string indicating the transformation used among `asinh` `biexp`, `log10` and `none`. Default is `asinh` transformation.
`num_col`	An integer vector of index indicating the columns to be transform. Default is `1:ncol(M)` to transform all the data.

Details

First of all, data can be transformed using different transformations. The algorithm is based on the construction of a binary tree, the nodes of which are subpopulations of cells. At each node, observed cells and markers are modeled by both a family of normal distributions and a family of bi-modal normal mixture distributions. Splitting is done according to a normalized difference of AIC between the two families.

Value

An object of class 'cytomeTree' providing a partitioning of the set of n cells.

- `annotation` A `data.frame` containing the annotation of each cell population underlying the tree pattern.
- `labels` The partitioning of the set of n cells.
- `M` The transformed matrix of mass cytometry.
- `mark_tree` A two level list containing markers used for node splitting.
- `transformation` Transformation used
- `num_col` Indexes of columns transformed

Author(s)

Anthony Devaux, Boris Hejblum

Examples

```
data(IMdata)

# dimension of data
dim(IMdata)

# given the size of the dataset, the code below can take several minutes to run

if(interactive()){
# Don't transform Time et Cell_length column
num_col <- 3:ncol(IMdata)

# Build Cytoftree binary tree
tree <- CytofTree(M = IMdata, minleaf = 1, t = 0.1, transformation = "asinh", num_col = num_col)

# Annotation
annot <- Annotation(tree, plot = FALSE, K2markers = colnames(IMdata))

# Provide subpopulations
annot$combinations
}
```

`CytomeTree` *Binary tree algorithm for cytometry data analysis.*

Description

Binary tree algorithm for cytometry data analysis.

Usage

```
CytomeTree(M, minleaf = 1, t = 0.1, verbose = TRUE, force_first_markers = NULL)
```

Arguments

`M`	A matrix of size n x p containing cytometry measures of n cells on p markers.
`minleaf`	An integer indicating the minimum number of cells per population. Default is `1`.
`t`	A real positive-or-null number used for comparison with the normalized AIC computed at each node of the tree. A higher value limits the height of the tree.
`verbose`	A logical controlling if a text progress bar is displayed during the execution of the algorithm. By default is TRUE.
`force_first_markers`	a vector of index to split the data on first. This argument is used in the semi-supervised setting, forcing the algorithm to consider those markers first, in the order they appear in this `force_first_markers` vector, and forcing the split at every node. Default is `NULL`, in which case the clustering algorithm is unsupervised.

Details

The algorithm is based on the construction of a binary tree, the nodes of which are subpopulations of cells. At each node, observed cells and markers are modeled by both a family of normal distributions and a family of bi-modal normal mixture distributions. Splitting is done according to a normalized difference of AIC between the two families.

Value

An object of class 'cytomeTree' providing a partitioning of the set of n cells.

- `annotation` A `data.frame` containing the annotation of each cell population underlying the tree pattern.
- `labels` The partitioning of the set of n cells.
- `M` The input matrix.
- `mark_tree` A two level list containing markers used for node splitting.

Author(s)

Chariff Alkhassim, Boris Hejblum

Examples

```
head(DLBCL)

# number of cell event
N <- nrow(DLBCL)

# Cell events
cellevents <- DLBCL[, c("FL1", "FL2", "FL4")]
```

```
# Manual partitioning of the set N (from FlowCAP-I)
manual_labels <- DLBCL[, "label"]


# Build the binary tree
Tree <- CytomeTree(cellevents, minleaf = 1, t=.1)


# Retreive the resulting partition of the set N
Tree_Partition <- Tree$labels


# Plot node distributions
par(mfrow=c(1, 2))
plot_nodes(Tree)

# Choose a node to plot
plot_nodes(Tree,"FL4.1")

# Plot a graph of the tree
par(mfrow=c(1,1))
plot_graph(Tree,edge.arrow.size=.3, Vcex =.5, vertex.size = 30)

# Run the annotation algorithm
Annot <- Annotation(Tree,plot=FALSE)
Annot$combinations


# Compare to the annotation gotten from the tree
Tree$annotation


# Example of sought phenotypes
# Variable in which sought phenotypes can be entered in the form of matrices.
phenotypes <- list()

# Sought phenotypes:
## FL2+ FL4-.
phenotypes[[1]] <- rbind(c("FL2", 1), c("FL4", 0))

## FL2- FL4+.
phenotypes[[2]] <- rbind(c("FL2", 0), c("FL4", 1))

## FL2+ FL4+.
phenotypes[[3]] <- rbind(c("FL2", 1), c("FL4", 1))

# Retreive cell populations found using Annotation.
PhenoInfos <- RetrievePops(Annot, phenotypes)
PhenoInfos$phenotypesinfo

# F-measure ignoring cells labeled 0 as in FlowCAP-I.
# Use FmeasureC() in any other case.
FmeasureC_no0(ref=manual_labels, pred=Tree_Partition)



if(interactive()){

# Scatterplots.
library(ggplot2)

# Ignoring cells labeled 0 as in FlowCAP-I.
rm_zeros <- which(!manual_labels)

# Building the data frame to scatter plot the data.
FL1 <- cellevents[-c(rm_zeros),"FL1"]
FL2 <- cellevents[-c(rm_zeros),"FL2"]
FL4 <- cellevents[-c(rm_zeros),"FL4"]
n <- length(FL1)
Labels <- c(manual_labels[-c(rm_zeros)]%%2+1, Tree_Partition[-c(rm_zeros)])
Labels <- as.factor(Labels)
method <- as.factor(c(rep("FlowCap-I",n),rep("CytomeTree",n)))

scatter_df <- data.frame("FL2" = FL2, "FL4" = FL4, "labels" = Labels, "method" = method)
p <- ggplot2::ggplot(scatter_df,  ggplot2::aes_string(x = "FL2", y = "FL4", colour = "labels")) +
 ggplot2::geom_point(alpha = 1,cex = 1) +
 ggplot2::scale_colour_manual(values = c("green","red","blue")) +
 ggplot2::facet_wrap(~ method) +
 ggplot2::theme_bw() +
 ggplot2::theme(legend.position="bottom")
p

}
```

DLBCL — *Diffuse large B-cell lymphoma data set from the FlowCAP-I challenge.*

Description

Diffuse large B-cell lymphoma data set from the FlowCAP-I challenge.

Usage

```
data(DLBCL)
```

Format

A data frame with 5524 cell events and 3 markers.

Source

http://flowcap.flowsite.org/

`FmeasureC` *C++ implementation of the F-measure computation*

Description

C++ implementation of the F-measure computation

Usage

```
FmeasureC(pred, ref)
```

Arguments

`pred`	vector of a predicted partition
`ref`	vector of a reference partition

Author(s)

Boris Hejblum

`FmeasureC_no0` *C++ implementation of the F-measure computation without the reference class labeled "0"*

Description

Aghaeepour in FlowCAP 1 ignore the reference class labeled "0"

Usage

```
FmeasureC_no0(pred, ref)
```

Arguments

`pred`	vector of a predicted partition
`ref`	vector of a reference partition

Author(s)

Boris Hejblum

`HIPC` *HIPC T cell data set from HIPC program, patient 12828. The data was analyzed and gated by Stanford.*

Description

HIPC T cell data set from HIPC program, patient 12828. The data was analyzed and gated by Stanford.

Usage

```
data(HIPC)
```

Format

A data frame with 33992 cell events and 6 markers.

Source

https://www.immuneprofiling.org/ https://www.immunespace.org/ https://www.immunespace.org/project/HIPC/Lyoplate/begin.view?pageId=study.DATA_ANALYSIS

`IMdata` *Influenza vaccine response dataset*

Description

A dataset containing 10,000 cells and 39 markers of mass cytometry subsampled from the sample SUB116516.478 from the study SDY478 by Mark Davis retrieved from ImmuneSpace

Usage

```
data(IMdata)
```

Format

A data frame with 10,000 rows and 39 variables:

Source

https://www.immunespace.org/project/Studies/SDY478/begin.view?

plot_cytopop *Plot the cell count for each population using CytomeTree.*

Description

Plot the cell count for each population using CytomeTree.

Usage

```
plot_cytopop(
  AnnotObj,
  nbpop = 10,
  mincount = 1,
  maxcount = NULL,
  y_axis = c("abs_count", "prop")
)
```

Arguments

AnnotObj	An object of class Annotation.
nbpop	Number indicating the maximum of population plotted. Default is `10`
mincount	Number indicating the minimum of cell count for the populations. Default is `1`.
maxcount	Number indicating the maximum of cell count for the populations. Default is `NULL` i.e no maximum selected.
y_axis	a character string either `"abs_count"` or `"prop"` indicating whether the absolute cell count or the relative populations proportions should be plotted. Default is `"abs_count"`.

Author(s)

Anthony Devaux, Boris Hejblum

Examples

```
# Run CytomeTree
data(DLBCL)
cellevents <- DLBCL[,c("FL1", "FL2", "FL4")]
Tree <- CytomeTree(cellevents, minleaf = 1, t=.1)
Annot <- Annotation(Tree,plot=FALSE)

# Plot the cell count
plot_cytopop(Annot)
```

plot_graph *Plot the binary tree built using CytomeTree.*

Description

Plot the binary tree built using CytomeTree.

Usage

```
plot_graph(CytomeTreeObj, Ecex = 1, Ecolor = 8, Vcex = 0.8, Vcolor = 0, ...)
```

Arguments

`CytomeTreeObj`	An object of class CytomeTree.
`Ecex`	Number indicating the amount by which text on the edges should be scaled. Default is `1`.
`Ecolor`	An integer or a string of character to color edges of the graph. Default is `8`.
`Vcex`	Number indicating the amount by which text in the vertices should be scaled. Default is `.8`.
`Vcolor`	A vector of class numeric or character to color vertices of the graph. Default is `0`.
`...`	additional arguments to be passed to `plot_graph`

Author(s)

Chariff Alkhassim

plot_nodes *Plot the distribution of the observed cells at each node of the binary tree built using CytomeTree.*

Description

Plot the distribution of the observed cells at each node of the binary tree built using CytomeTree.

Usage

```
plot_nodes(
  CytomeTreeObj,
  nodes = NULL,
  nodesPerCol = NULL,
  nodesPerRow = NULL,
  ...
)
```

Arguments

`CytomeTreeObj`	An object of class CytomeTree.
`nodes`	A list of character elements containing the name of the nodes for which the distribution is to be plotted. Default is `NULL`, and plots the distribution for each node.
`nodesPerCol`	an integer specifying the number of plots to be displayed per column when plotting multiple nodes at once. Default is `NULL`.
`nodesPerRow`	an integer specifying the number of plots to be displayed per row when plotting multiple nodes at once. Default is `NULL`.
`...`	further arguments to be passed to `plot_grid`.

Details

if both `nodesPerCol` and `nodesPerRow` are `NULL` then all the nodes are plotted on a single page.

Value

a list of `ggplot2` plot objects, containing each node plot.

Author(s)

Chariff Alkhassim, Boris Hejblum

Examples

```
data(DLBCL)
myct <- CytomeTree(DLBCL[, c("FL1", "FL2", "FL4")], minleaf = 1, t=.1)
plot_nodes(myct)
```

`RetrievePops` *Retrieve cell populations found using Annotation.*

Description

Retrieve cell populations found using Annotation.

Usage

```
RetrievePops(AnnotationObj, phenotypes)
```

Arguments

`AnnotationObj`	An object of class Annotation.
`phenotypes`	A list containing at least one element of class matrix describing a sought phenotype. Each matrix should have two columns where the name of a used marker is associated to a value chosen between 0, 1 and 2. 0 translates to -, 1 to + and 2 to ++.

Value

A `list` of two elements.

- `phenotypesinfo` A `list` containing informations about sought populations.
- `Mergedleaves` The partitioning of the set of n cells with potentially merged leaves.

Author(s)

Chariff Alkhassim, Boris Hejblum

Index