Package 'CytoDx'

February 5, 2025

Type Package

Title Robust prediction of clinical outcomes using cytometry data without cell gating

Version 1.27.0

Author Zicheng Hu

Maintainer Zicheng Hu <zicheng.hu@ucsf.edu>

Description This package provides functions that predict clinical outcomes using single cell data (such as flow cytometry data, RNA single cell sequencing data) without the requirement of cell gating or clustering.

License GPL-2

Encoding UTF-8

LazyData true

Imports doParallel, dplyr, glmnet, rpart, rpart.plot, stats, flowCore,grDevices, graphics, utils

Depends R (>= 3.5)

Suggests knitr, rmarkdown

VignetteBuilder knitr, rmarkdown

RoxygenNote 6.1.0

biocViews ImmunoOncology, CellBiology, FlowCytometry, StatisticalMethod, Software, CellBasedAssays, Regression, Classification, Survival

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

git_branch devel

git_last_commit 05557f1

git_last_commit_date 2024-10-29

Repository Bioconductor 3.21

Date/Publication 2025-02-05

Contents

`CytoDx.fit` *Build the CytoDx model*

Description

A function that builds the CytoDx model.

Usage

```
CytoDx.fit(x, y, xSample, family = c("gaussian", "binomial", "poisson",
  "multinomial", "cox", "mgaussian"), type1 = "response",
  type2 = "response", parallelCore = 1, reg = FALSE, ...)
```

Arguments

`x`	The marker profile of cells pooled from all samples. Each row is a cell, each column is a marker.
`y`	The clinical outcomes associated with samples to which cells belong. Length must be equal to nrow(x). For family="binomial" should be either a factor with two levels, or a two-column matrix of counts or proportions (the second column is treated as the target class; for a factor, the last level in alphabetical order is the target class). For family="multinomial", can be a nc>=2 level factor, or a matrix with nc columns of counts or proportions. For either "binomial" or "multinomial", if y is presented as a vector, it will be coerced into a factor. For family="cox", y should be a two-column matrix with columns named 'time' and 'status'. The latter is a binary variable, with '1' indicating death, and '0' indicating right censored. The function Surv() in package survival produces such a matrix. For family="mgaussian", y is a matrix of quantitative responses.
`xSample`	A vector specifying which sample each cell belongs to. Length must equal to nrow(x).
`family`	Response type. Must be one of the following: "gaussian","binomial","poisson","multinomial","cox","mga

type1	Type of first level prediction. Type of prediction required. Type "link" gives the linear predictors for "binomial", "multinomial", "poisson" or "cox" models; for "gaussian" models it gives the fitted values. Type "response" gives the fitted probabilities for "binomial" or "multinomial", fitted mean for "poisson" and the fitted relative-risk for "cox"; for "gaussian" type "response" is equivalent to type "link".
type2	Type of second level prediction.
parallelCore	The number of core to be used. Only used when reg is TRUE.
reg	If elestic net regularization will be used.
...	Other parameters to be passed into the glmnet or the cv.glmnet function in the glmnet package.

Value

Returns a list. train.Data.cell contains the trainig data and the predicted y for the training data at the cell level. model.cell contains the cell stage statistical model. Data.sample contains the trainig data and the predicted y for the training data at the sample level. model.sample contains the sample stage statistical model. family specifies the regression type. method specifies the type of learning method. type.cell is the type of cell level prediction. type.sample is the type of sample level prediction.

Examples

```
# Find the table containing fcs file names in CytoDx package
path <- system.file("extdata",package="CytoDx")
# read the table
fcs_info <- read.csv(file.path(path,"fcs_info.csv"))
# Specify the path to the cytometry files
fn <- file.path(path,fcs_info$fcsName)
# Read cytometry files using fcs2DF function
train_data <- fcs2DF(fcsFiles=fn,
                     y=fcs_info$Label,
                     assay="FCM",
                     b=1/150,
                     excludeTransformParameters=
                       c("FSC-A","FSC-W","FSC-H","Time"))
# build the model
fit <- CytoDx.fit(x=as.matrix(train_data[,1:7]),
                  y=train_data$y,
                  xSample = train_data$xSample,
                  reg=FALSE,
                  family="binomial")
# check accuracy for training data
pred <- CytoDx.pred(fit,
                    xNew=as.matrix(train_data[,1:7]),
                    xSampleNew=train_data$xSample)

boxplot(pred$xNew.Pred.sample$y.Pred.s0~
          fcs_info$Label)
```

CytoDx.pred *Make prediction using the CytoDx model*

Description

A function that makes prediction using the CytoDx model.

Usage

```
CytoDx.pred(fit, xNew, xSampleNew)
```

Arguments

fit	The two stage statistical model. Must be the object returned by CytoDx.fit.
xNew	The marker profile of cells pooled from all new samples. Each row is a cell, each column is a marker.
xSampleNew	A vector specifying which sample each cell belongs to. Length must equal to nrow(xNew).

Value

Returns a list. xNew.Pred1 contains the predicted y for the new data at the cell level. xNew.Pred2 contains the predicted y for the new data at the sample level.

Examples

```
# Find the table containing fcs file names in CytoDx package
path <- system.file("extdata",package="CytoDx")
# read the table
fcs_info <- read.csv(file.path(path,"fcs_info.csv"))
# Specify the path to the cytometry files
fn <- file.path(path,fcs_info$fcsName)
train_data <- fcs2DF(fcsFiles=fn,
                     y=fcs_info$Label,
                     assay="FCM",
                     b=1/150,
                     excludeTransformParameters=
                       c("FSC-A","FSC-W","FSC-H","Time"))
# build the model
fit <- CytoDx.fit(x=as.matrix(train_data[,1:7]),
                  y=train_data$y,
                  xSample = train_data$xSample,
                  reg=FALSE,
                  family="binomial")
# check accuracy for training data
pred <- CytoDx.pred(fit,
                    xNew=as.matrix(train_data[,1:7]),
                    xSampleNew=train_data$xSample)
```

```
boxplot(pred$xNew.Pred.sample$y.Pred.s0~
          fcs_info$Label)
```

`fcs2DF` *Convert fcs files to a data frame*

Description

A function that convert fcs files to a data frame.

Usage

```
fcs2DF(fcsFiles, y = NULL, assay = c("FCM", "CyTOF"), b = 1/200,
  fileSampleSize = 5000, compFiles = NULL, nameDict = NULL,
  excludeTransformParameters = c("FSC-A", "FSC-W", "FSC-H", "Time",
  "Cell_length"))
```

Arguments

`fcsFiles`	A vector specifying the location of fcs files (relative to working directory).
`y`	A vector containing the clinical outcome of each sample. Must have the same length as fcsFiles. Null for testing data.
`assay`	Either "FCM" or "CyTOF" to indicate the type of cytometry data.
`b`	A positive number used to specify the arcsinh transformation. f(x) = asinh (b*x) where x is the original value and f(x) is the value after transformation. The suggested value is 1/150 for flow cytometry (FCM) data and 1/8 for CyTOF data.
`fileSampleSize`	An integer specifying the number of events sampled from each fcs file. If NULL, all the events will be pre-processed and wrote out to the new fcs files.
`compFiles`	A vector specifying the paths of user supplied compensation matrix for each fcs file. The matrix must be stored in csv files.
`nameDict`	A vector used to change marker names.Each element in the vector is the prefered name of a marker. The name of each element is the marker name used in the fcs file. For example, a vector c("CD8b"="CD8","cd8"="CD8") will change "CD8b" and "cd8" into "CD8", making annotations more consistent.
`excludeTransformParameters`	A vector specifying the name of parameters not to be transformed (left at linear scale).

Value

Returns a data frame containing the preprocessed cytometry data. Cells from different fcs files are combined into one flow frame. A new column, xSample, is introduced to indicate the origin of each cell. The data frame also includes the clinical outcome y.

Examples

```
# Find the table containing fcs file names in CytoDx package
path <- system.file("extdata",package="CytoDx")
# read the table
fcs_info <- read.csv(file.path(path,"fcs_info.csv"))
# Specify the path to the cytometry files
fn <- file.path(path,fcs_info$fcsName)
# Read cytometry files using fcs2DF function
train_data <- fcs2DF(fcsFiles=fn,
                     y=fcs_info$Label,
                     assay="FCM",
                     b=1/150,
                     excludeTransformParameters=
                       c("FSC-A","FSC-W","FSC-H","Time"))
```

---

meanUnique *Calulate mean or take unique elements of a vector*

Description

A function that calulate mean or take unique elements of a vector.

Usage

```
meanUnique(x)
```

Arguments

x	a vector

Value

If x is numeric, returns the mean. Otherwise, returns the unique elements of x.

Examples

```
x <- 1:5
meanUnique(x)
x=c("a","a","b")
meanUnique(x)
```

pRank *Percentile rank transformation of the data*

Description

A function that performs the rank transformation of the data.

Usage

```
pRank(x, xSample)
```

Arguments

x	A data frame containing the pooled data from fcs files. Each row is a cell, each column is a marker.
xSample	A vector specifying which sample each cell belongs to. Length must equal to nrow(x).

Value

Returns data frame containing rank transformed data.

Examples

```
x <- pRank(x=iris[,1:4],xSample=iris$Species)
```

rank.ub.average *Percentile rank transformation of a vector*

Description

A function that performs the Percentile rank transformation of a vector

Usage

```
rank.ub.average(x)
```

Arguments

x	A numeric vector.

Value

Returns the percentile rank of each element.

Examples

```
rank.ub.average(1:10)
```

set2DF *convert a flowSet to a data frame*

Description

A function that convert a flowSet to a data frame.

Usage

```
set2DF(flowSet, fcsFiles, y = NULL)
```

Arguments

flowSet	A flowSet object
fcsFiles	A vector containing the name of each fcs file included in flowSet.
y	The clinical outcome each fcs file associated with. Null for testing data.

Value

Returns a data frame containing the cytometry data. Cells from different fcs files are combined into one flow frame. A new column, xSample, is introduced to indicate the origin of each cell. The data frame also includes the clinical outcome y.

Examples

```
library(flowCore)
# Find the table containing fcs file names in CytoDx package
path <- system.file("extdata",package="CytoDx")
# read the table
fcs_info <- read.csv(file.path(path,"fcs_info.csv"))
# Specify the path to the cytometry files
fn <- file.path(path,fcs_info$fcsName)
fSet <- read.flowSet(fn)
df <- set2DF(flowSet=fSet,fcsFiles=fn,y = fcs_info$Label)
```

treeGate *Use decision tree to find a group of cells that are associated with clinical outcome.*

Description

A function that sse decision tree to find a group of cells that are associated with clinical outcome.

Usage

```
treeGate(P, x, ...)
```

Arguments

P	The predicted association of each cell with a clinical outcome.
x	The marker profile of each cell. Each row is a cell, each column is a marker. Must have length(P) rows.
...	Other parameters to be passed into the rpart function

Value

Returns a object created by rpart function. Also plots a graph of decision tree.

Examples

```
# Find the table containing fcs file names in CytoDx package
path=system.file("extdata",package="CytoDx")
# read the table
fcs_info <- read.csv(file.path(path,"fcs_info.csv"))
# Specify the path to the cytometry files
fn <- file.path(path,fcs_info$fcsName)
# Read cytometry files using fcs2DF function
train_data <- fcs2DF(fcsFiles=fn,
                     y=fcs_info$Label,
                     assay="FCM",
                     b=1/150,
                     excludeTransformParameters=
                       c("FSC-A","FSC-W","FSC-H","Time"))
# build the model
fit <- CytoDx.fit(x=as.matrix(train_data[,1:7]),
                  y=train_data$y,
                  xSample = train_data$xSample,
                  reg=FALSE,
                  family="binomial")
# check accuracy for training data
pred <- CytoDx.pred(fit,
                    xNew=as.matrix(train_data[,1:7]),
                    xSampleNew=train_data$xSample)

boxplot(pred$xNew.Pred.sample$y.Pred.s0~
          fcs_info$Label)

# Find the associated population using treeGate
TG <- treeGate(P = fit$train.Data.cell$y.Pred.s0,
               x= train_data[,1:7])
```

Index