Package 'SCFA'

July 14, 2025

Type Package

Title SCFA: Subtyping via Consensus Factor Analysis

Version 1.19.0

Description Subtyping via Consensus Factor Analysis (SCFA) can efficiently remove noisy signals from consistent molecular patterns in multi-omics data.
SCFA first uses an autoencoder to select only important features and then repeatedly performs factor analysis to represent the data with different numbers of factors.
Using these representations, it can reliably identify cancer subtypes and accurately predict risk scores of patients.

License LGPL

Encoding UTF-8

LazyData true

Depends R (>= 4.0)

Imports matrixStats, BiocParallel, torch (>= 0.3.0), coro, igraph, Matrix, cluster, psych, glmnet, RhpcBLASctl, stats, utils, methods, survival

RoxygenNote 7.1.1

biocViews Survival, Clustering, Classification

Suggests knitr, rmarkdown, BiocStyle

VignetteBuilder knitr

URL https://github.com/duct317/SCFA

BugReports https://github.com/duct317/SCFA/issues

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

git_branch devel

git_last_commit 29eaf0a

git_last_commit_date 2025-04-15

Repository Bioconductor 3.22

Date/Publication 2025-07-13

Author Duc Tran [aut, cre], Hung Nguyen [aut], Tin Nguyen [fnd]

Maintainer Duc Tran <duct@nevada.unr.edu>

Contents

GBM *GBM*

Description

GBM dataset, including microRNA and survidal data.

Usage

```
GBM
```

Format

A list with two items:

data List of microRNA data matrix.

survival Survival information.

SCFA *SCFA*

Description

The main function to perform subtyping. It takes a list of data matrices as the input and outputs the subtype for each patient

Usage

```
SCFA(dataList, k = NULL, max.k = 5, ncores = 10L, seed = NULL)
```

Arguments

`dataList`	List of data matrices. In each matrix, rows represent samples and columns represent genes/features.
`k`	Number of clusters, leave as default for auto detection.
`max.k`	Maximum number of cluster
`ncores`	Number of processor cores to use.
`seed`	Seed for reproducibility, you still need to use set.seed function for full reproducibility.

Value

A numeric vector containing cluster assignment for each sample.

Examples

```
#Load example data (GBM dataset)
data("GBM")
#List of one matrix (microRNA data)
dataList <- GBM$data
#Survival information
survival <- GBM$survival
library(survival)
#Generating subtyping result
set.seed(1)
subtype <- SCFA(dataList, seed = 1, ncores = 2L)
#Perform survival analysis on the result
coxFit <- coxph(Surv(time = Survival, event = Death) ~ as.factor(subtype), data = survival, ties="exact")
coxP <- round(summary(coxFit)$sctest[3],digits = 20)
print(coxP)
```

SCFA.class *SCFA.class*

Description

Perform risk score prediction on input data. This function requires training data with survival information. The output is the risk scores of patients in testing set.

Usage

```
SCFA.class(dataListTrain, trainLabel, dataListTest, ncores = 10L, seed = NULL)
```

Arguments

dataListTrain	List of training data matrices. In each matrix, rows represent samples and columns represent genes/features.
trainLabel	Survival information of patient in training set in form of Surv object.
dataListTest	List of testing data matrices. In each matrix, rows represent samples and columns represent genes/features.
ncores	Number of processor cores to use.
seed	Seed for reproducibility, you still need to use set.seed function for full reproducibility.

Value

A vector of risk score predictions for patient in test set.

Examples

```
#Load example data (GBM dataset)
data("GBM")
#List of one matrix (microRNA data)
dataList <- GBM$data
#Survival information
survival <- GBM$survival
library(survival)
```

```
#Split data to train and test
set.seed(1)
idx <- sample.int(nrow(dataList[[1]]), round(nrow(dataList[[1]])/2) )
survival$Survival <- survival$Survival - min(survival$Survival) + 1 # Survival time must be positive
trainList <- lapply(dataList, function(x) x[idx, ] )
trainSurvival <- Surv(time = survival[idx,]$Survival, event =  survival[idx,]$Death)
testList <- lapply(dataList, function(x) x[-idx, ] )
testSurvival <- Surv(time = survival[-idx,]$Survival, event =  survival[-idx,]$Death)
#Perform risk prediction
result <- SCFA.class(trainList, trainSurvival, testList, seed = 1, ncores = 2L)
#Validation using concordance index
c.index <- concordance(coxph(testSurvival ~ result))$concordance
print(c.index)
```

Index