

Package ‘M3C’

November 5, 2024

Title Monte Carlo Reference-based Consensus Clustering

Version 1.29.0

Author Christopher John, David Watson

Maintainer Christopher John <chris.r.john86@gmail.com>

Description M3C is a consensus clustering algorithm that uses a Monte Carlo simulation to eliminate overestimation of K and can reject the null hypothesis $K=1$.

Depends R ($\geq 3.5.0$)

License AGPL-3

Encoding UTF-8

LazyData true

Imports ggplot2, Matrix, doSNOW, cluster, parallel, foreach, doParallel, matrixcalc, Rtsne, corpcor, umap

Suggests knitr, rmarkdown

VignetteBuilder knitr

RoxygenNote 7.0.1

biocViews Clustering, GeneExpression, Transcription, RNASeq, Sequencing, ImmunoOncology

git_url <https://git.bioconductor.org/packages/M3C>

git_branch devel

git_last_commit cbbb878

git_last_commit_date 2024-10-29

Repository Bioconductor 3.21

Date/Publication 2024-11-04

Contents

clustersim	2
desx	3
featurefilter	3

M3C	4
mydata	6
pca	6
tsne	7
umap	8
Index	10

clustersim	<i>clustersim: A cluster simulator for testing clustering algorithms</i>
------------	--

Description

clustersim: A cluster simulator for testing clustering algorithms

Usage

```
clustersim(n, n2, r, K, alpha, wobble, redp = NULL, print = FALSE,
  seed = NULL)
```

Arguments

n	Numerical value: The number of samples, it must be square rootable
n2	Numerical value: The number of features
r	Numerical value: The radius to define the initial circle (use approx n/100)
K	Numerical value: How many clusters to simulate
alpha	Numerical value: How far to pull apart the clusters
wobble	Numerical value: The degree of noise to add to the sample co ordinates
redp	Numerical value: The fraction of samples to remove from one cluster
print	Logical flag: whether to print the PCA into current directory
seed	Numerical value: fixes the seed if you want to repeat results

Value

A list: containing 1) matrix with simulated data in it

Examples

```
res <- clustersim(225, 900, 8, 4, 0.75, 0.025, redp = NULL, seed=123)
```

desx	<i>GBM clinical annotation data</i>
------	-------------------------------------

Description

This is the clinical annotation data from the GBM dataset, it contains the class of the tumour which is one of: classical, mesenchymal, neural, proneural. It is a data frame with 2 columns and 50 rows.

Author(s)

Chris John <chris.r.john86@gmail.com>

References

Verhaak, Roel GW, et al. "Integrated genomic analysis identifies clinically relevant subtypes of glioblastoma characterized by abnormalities in PDGFRA, IDH1, EGFR, and NF1." *Cancer cell* 17.1 (2010): 98-110.

featurefilter	<i>featurefilter: A function for filtering features</i>
---------------	---

Description

This function is to filter features based on variance. Depending on the data different metrics will be more appropriate, simple variance is included if variance does not tend to increase with the mean. There is also the median absolute deviation which is a more robust metric than variance, this is preferable. The coefficient of variation (A) or its second order derivative (A2) (Kvalseth, 2017) are also included which standardise the standard deviation with respect to the mean. It is best to manually examine the mean-variance relationship of the data, for example, using the results from this function together with the `qplot` function from `ggplot2`.

Usage

```
featurefilter(mydata, percentile = 10, method = "MAD", topN = 20)
```

Arguments

mydata	Data frame: should have samples as columns and rows as features
percentile	Numerical value: the top X percent most variable features should be kept
method	Character vector: variance (var), coefficient of variation (A), second order A (A2), median absolute deviation (MAD)
topN	Numerical value: the number of most variable features to display

Value

A list, containing: 1) filtered data 2) statistics for each feature order according to the defined filtering metric

References

Kvålseth, Tarald O. "Coefficient of variation: the second-order alternative." *Journal of Applied Statistics* 44.3 (2017): 402-415.

Examples

```
filtered <- featurefilter(mydata,percentile=10)
```

M3C

M3C: Monte Carlo Reference-based Consensus Clustering

Description

This is the M3C core function, which is a reference-based consensus clustering algorithm. The basic idea is to use a multi-core enabled Monte Carlo simulation to drive the creation of a null distribution of stability scores. The Monte Carlo simulations maintains the feature correlation structure of the input data. Then the null distribution is used to compare the reference scores with the real scores and an empirical p value is calculated for every value of K to test the null hypothesis K=1. We derive the Relative Cluster Stability Index (RCSI) as a metric for selecting K, which is based on a comparison against the reference mean. A fast alternative is also included that includes a penalty term to prevent overestimation of K, we call regularised consensus clustering.

Usage

```
M3C(mydata, cores = 1, iters = 25, maxK = 10, pItem = 0.8,
    des = NULL, ref_method = c("reverse-pca", "chol"), repsref = 100,
    repsreal = 100, clusteralg = c("pam", "km", "spectral", "hc"),
    pacx1 = 0.1, pacx2 = 0.9, seed = 123, objective = "entropy",
    removeplots = FALSE, silent = FALSE, fsize = 18, method = 1,
    lambda default = 0.1, tunelambda = TRUE, lseq = seq(0.02, 0.1, by =
    0.02), lthick = 2, dotsize = 3)
```

Arguments

mydata	Data frame or matrix: Contains the data, with samples as columns and rows as features
cores	Numerical value: how many cores to split the monte carlo simulation over
iters	Numerical value: how many Monte Carlo iterations to perform (default: 25, recommended: 5-100)
maxK	Numerical value: the maximum number of clusters to test for, K (default: 10)
pItem	Numerical value: the fraction of points to resample each iteration (default: 0.8)

<code>des</code>	Data frame: contains annotation data for the input data for automatic reordering
<code>ref_method</code>	Character string: refers to which reference method to use
<code>repsref</code>	Numerical value: how many resampling reps to use for reference (default: 100, recommended: 100-250)
<code>repsreal</code>	Numerical value: how many resampling reps to use for real data (default: 100, recommended: 100-250)
<code>clusteralg</code>	String: dictates which inner clustering algorithm to use (default: PAM)
<code>pacx1</code>	Numerical value: The 1st x co-ordinate for calculating the pac score from the CDF (default: 0.1)
<code>pacx2</code>	Numerical value: The 2nd x co-ordinate for calculating the pac score from the CDF (default: 0.9)
<code>seed</code>	Numerical value: specifies seed, set to NULL for different results each time
<code>objective</code>	Character string: whether to use 'PAC' or 'entropy' objective function (default = entropy)
<code>removeplots</code>	Logical flag: whether to remove all plots from view
<code>silent</code>	Logical flag: whether to remove messages or not
<code>fsize</code>	Numerical value: determines the font size of the ggplot2 plots
<code>method</code>	Numerical value: 1 refers to the Monte Carlo simulation method, 2 to regularised consensus clustering
<code>lambddefault</code>	Numerical value: if not tuning fixes the default (default: 0.1)
<code>tunelambda</code>	Logical flag: whether to tune lambda or not
<code>lseq</code>	Numerical vector: vector of lambda values to tune over (default = seq(0.05,0.1,by=0.01))
<code>lthick</code>	Numerical value: determines the line thickness of the ggplot2 plot
<code>dotsize</code>	Numerical value: determines the dotsize of the ggplot2 plot

Value

A list, containing: 1) the stability results and 2) all the output data (another list) 3) reference stability scores (see vignette for more details on how to easily access)

Examples

```
res <- M3C(mydata)
```

mydata *GBM expression data*

Description

This is the expression data from the GBM dataset. It is a data frame with 50 columns and 1740 rows.

Author(s)

Chris John <chris.r.john86@gmail.com>

References

Verhaak, Roel GW, et al. "Integrated genomic analysis identifies clinically relevant subtypes of glioblastoma characterized by abnormalities in PDGFRA, IDH1, EGFR, and NF1." *Cancer cell* 17.1 (2010): 98-110.

pca *pca: A principal component analysis function*

Description

This is a flexible PCA function that can be run on a standard data frame. It is a wrapper for prcomp/ggplot2 code and can be customised with different colours and font sizes and more.

Usage

```
pca(mydata, printres = FALSE, labels = FALSE, text = FALSE,
    axistextsize = 18, legendtextsize = 18, dotsize = 5,
    textlabelsize = 4, legendtitle = "Group", controlscale = FALSE,
    scale = 1, low = "grey", high = "red", colvec = c("skyblue",
    "gold", "violet", "darkorchid", "slateblue", "forestgreen", "violetred",
    "orange", "midnightblue", "grey31", "black"), printheight = 20,
    printwidth = 22, pcx = 1, pcy = 2, scaler = FALSE)
```

Arguments

mydata	Data frame or matrix: if dataframe/matrix should have samples as columns and rows as features
printres	Logical flag: whether to print the PCA into current directory
labels	Character vector: if we want to just label with gender for example
text	Character vector: if we wanted to label the samples with text IDs to look for outliers

<code>axistextsize</code>	Numerical value: axis text size
<code>legendtextsize</code>	Numerical value: legend text size
<code>dotsize</code>	Numerical value: dot size
<code>textlabelsize</code>	Numerical value: text inside plot label size
<code>legendtitle</code>	Character vector: text legend title
<code>controlscale</code>	Logical flag: whether to control the colour scale
<code>scale</code>	Numerical value: 1=spectral palette, 2>manual low and high palette, 3=categorical labels
<code>low</code>	Character vector: continuous scale low colour
<code>high</code>	Character vector: continuous scale high colour
<code>colvec</code>	Character vector: a series of colours in vector for categorical labels, e.g. <code>c("sky blue", "gold")</code>
<code>printheight</code>	Numerical value: png height (default=20)
<code>printwidth</code>	Numerical value: png width (default=22)
<code>pcx</code>	Numerical value: which PC to plot on X axis (default=1)
<code>pcy</code>	Numerical value: which PC to plot on Y axis (default=2)
<code>scaler</code>	Logical flag: whether to scale the features of the input data (rows) (default=FALSE)

Value

A PCA plot object

Examples

```
PCA <- pca(mydata)
```

<code>tsne</code>	<i>tsne: A t-SNE function</i>
-------------------	-------------------------------

Description

This is a flexible t-SNE function that can be run on a standard data frame. It is a wrapper for `Rtsne/ggplot2` code and can be customised with different colours and font sizes and more.

Usage

```
tsne(mydata, labels = FALSE, perplex = 15, printres = FALSE,
      seed = FALSE, axistextsize = 18, legendtextsize = 18,
      dotsize = 5, textlabelsize = 4, legendtitle = "Group",
      controlscale = FALSE, scale = 1, low = "grey", high = "red",
      colvec = c("skyblue", "gold", "violet", "darkorchid", "slateblue",
                "forestgreen", "violetred", "orange", "midnightblue", "grey31", "black"),
      printheight = 20, printwidth = 22, text = FALSE)
```

Arguments

mydata	Data frame or matrix: if dataframe/matrix should have samples as columns and rows as features
labels	Character vector: if we want to just label with gender for example
perplex	Numerical value: perplexity value that Rtsne uses internally
printres	Logical flag: whether to print the t-SNE into current directory
seed	Numerical value: optionally set the seed
axistextsize	Numerical value: axis text size
legendtextsize	Numerical value: legend text size
dotsize	Numerical value: dot size
textlabelsize	Numerical value: text inside plot label size
legendtitle	Character vector: text legend title
controlscale	Logical flag: whether to control the colour scale
scale	Numerical value: 1=spectral palette, 2>manual low and high palette, 3=categorical labels
low	Character vector: continuous scale low colour
high	Character vector: continuous scale high colour
colvec	Character vector: a series of colours in vector for categorical labels, e.g. <code>c("sky blue", "gold")</code>
printheight	Numerical value: png height
printwidth	Numerical value: png width
text	Character vector: if we wanted to label the samples with text IDs to look for outliers

Value

A t-SNE plot object

Examples

```
TSNE <- tsne(mydata,perplex=15)
```

umap

umap: A umap function

Description

This is a flexible umap function that can be run on a standard data frame. It is a wrapper for umap/ggplot2 code and can be customised with different colours and font sizes and more.

Usage

```
umap(mydata, labels = FALSE, printres = FALSE, seed = FALSE,
     axistextsize = 18, legendtextsize = 18, dotsize = 5,
     textlabelsize = 4, legendtitle = "Group", controlscale = FALSE,
     scale = 1, low = "grey", high = "red", colvec = c("skyblue",
     "gold", "violet", "darkorchid", "slateblue", "forestgreen", "violetred",
     "orange", "midnightblue", "grey31", "black"), printheight = 20,
     printwidth = 22, text = FALSE)
```

Arguments

mydata	Data frame or matrix: if dataframe/matrix should have samples as columns and rows as features
labels	Character vector: if we want to just label with gender for example
printres	Logical flag: whether to print the UMAP into current directory
seed	Numerical value: optionally set the seed
axistextsize	Numerical value: axis text size
legendtextsize	Numerical value: legend text size
dotsize	Numerical value: dot size
textlabelsize	Numerical value: text inside plot label size
legendtitle	Character vector: text legend title
controlscale	Logical flag: whether to control the colour scale
scale	Numerical value: 1=spectral palette, 2>manual low and high palette, 3=categorical labels
low	Character vector: continuous scale low colour
high	Character vector: continuous scale high colour
colvec	Character vector: a series of colours in vector for categorical labels, e.g. c("sky blue", "gold")
printheight	Numerical value: png height
printwidth	Numerical value: png width
text	Character vector: if we wanted to label the samples with text IDs to look for outliers

Value

A umap plot object

Examples

```
UMAP <- umap(mydata)
```

Index

* data

desx, 3

mydata, 6

clustersim, 2

desx, 3

featurefilter, 3

M3C, 4

mydata, 6

pca, 6

tsne, 7

umap, 8