Package 'multiClust'

January 23, 2022

Type Package

Title multiClust: An R-package for Identifying Biologically Relevant Clusters in Cancer Transcriptome Profiles

Version 1.24.0

Date 2021-07-27

Description Clustering is carried out to identify patterns in transcriptomics profiles to determine clinically relevant subgroups of patients. Feature (gene) selection is a critical and an integral part of the process. Currently, there are many feature selection and clustering methods to identify the relevant genes and perform clustering of samples. However, choosing an appropriate methodology is difficult. In addition, extensive feature selection methods have not been supported by the available packages. Hence, we developed an integrative R-package called multiClust that allows researchers to experiment with the choice of combination of methods for gene selection and clustering with ease. Using multiClust, we identified the best performing clustering methodology in the context of clinical outcome. Our observations demonstrate that simple methods such as variance-based ranking perform well on the majority of data sets, provided that the appropriate number of genes is selected. However, different gene ranking and selection methods remain relevant as no methodology works for all studies.

License GPL (>= 2)

biocViews FeatureExtraction, Clustering, GeneExpression, Survival

LazyData TRUE

Imports mclust, ctc, survival, cluster, dendextend, amap, graphics, grDevices

Suggests knitr, rmarkdown, gplots, RUnit, BiocGenerics, preprocessCore, Biobase, GEOquery

VignetteBuilder knitr

RoxygenNote 5.0.0

NeedsCompilation no

Author Nathan Lawlor [aut, cre], Peiyong Guan [aut], Alec Fabbri [aut], Krish Karuturi [aut], Joshy George [aut]

Maintainer Nathan Lawlor <nathan.lawlor03@gmail.com>

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

git_branch RELEASE_3_14

git_last_commit a435bdc

git_last_commit_date 2021-10-26

Date/Publication 2022-01-23

R topics documented:

`avg_probe_exp` *Function to produce a matrix containing the average expression of each gene probe within each sample cluster.*

Description

Function to produce a matrix containing the average expression of each gene probe within each sample cluster.

Usage

```
avg_probe_exp(sel.exp, samp_cluster, data_name, cluster_type = "HClust",
  distance = "euclidean", linkage_type = "ward.D2",
  probe_rank = "SD_Rank", probe_num_selection = "Fixed_Probe_Num",
  cluster_num_selection = "Fixed_Clust_Num")
```

Arguments

`sel.exp`	Object containing the numeric selected gene expression matrix. This object is an output of the probe_ranking function.
`samp_cluster`	Object vector containing the samples and the cluster number they belong to. This is an output of the cluster_analysis function.
`data_name`	String indicating the cancer type and name of the dataset being analyzed. This name will be used to label the sample dendrograms and heatmap files.

`cluster_type` String indicating the type of clustering method used in the cluster_analysis function. "Kmeans" or "HClust" are the two options.

`distance` String describing the distance metric uses for HClust in the cluster_analysis function. Options include one of "euclidean", "maximum", manhattan", "canberra", "binary", or "minkowski".

`linkage_type` String describing the linkage metric used in the cluster_analysis function. Options include "ward.D2", "average", "complete", "median", "centroid", "single", and "mcquitty".

`probe_rank` String indicating the feature selection method used in the probe_ranking function. Options include "CV_Rank", "CV_Guided", "SD_Rank", and "Poly".

`probe_num_selection` String indicating the way in which probes were selected in the number_probes function. Options include "Fixed_Probe_Num", "Percent_Probe_Num", and "Adaptive_Probe_Num".

`cluster_num_selection` String indicating how the number of clusters were determined in the number_clusters function. Options include "Fixed_Clust_Num" and "Gap_Statistic".

Value

Returns an object matrix with the average mean expression for each probe in each sample cluster. Also outputs the object to a text file.

Author(s)

Nathan Lawlor, Alec Fabbri

See Also

number_clusters, number_probes, probe_ranking, cluster_analysis

Examples

```
# Produce matrix of average expression of each probe in each cluster
# Load in a data file
data_file <- system.file("extdata", "GSE2034.normalized.expression.txt",
    package="multiClust")
data <- input_file(input=data_file)
# Choose 300 genes to select for
gene_num <- number_probes(input=data_file, data.exp=data, Fixed=300,
   Percent=NULL, Adaptive=NULL)
# Choose the "CV_Rank" Method for gene ranking
sel.data <- probe_ranking(input=data_file, probe_number=300,
    probe_num_selection="Fixed_Probe_Num", data.exp=data, method="CV_Rank")
# Choose a fixed cluster number of 3
clust_num <- number_clusters(data.exp=data, Fixed=3, gap_statistic=NULL)
# Call function for Kmeans parameters
kmeans_analysis <- cluster_analysis(sel.exp=sel.data, cluster_type="Kmeans",
    distance=NULL, linkage_type=NULL, gene_distance=NULL,
```

```
    num_clusters=3, data_name="GSE2034 Breast",
    probe_rank="CV_Rank", probe_num_selection="Fixed_Probe_Num",
    cluster_num_selection="Fixed_Clust_Num")
# Call function for average matrix expression calculation
avg_matrix <- avg_probe_exp(sel.exp=sel.data, samp_cluster=kmeans_analysis,
    data_name="GSE2034 Breast", cluster_type="Kmeans", distance=NULL,
    linkage_type=NULL, probe_rank="CV_Rank",
    probe_num_selection="Fixed", cluster_num_selection="Fixed_Clust_Num")
```

`cluster_analysis` *Function to perform Kmeans or Hierarchical clustering analysis of the selected gene probe expression data.*

Description

Function to perform Kmeans or Hierarchical clustering analysis of the selected gene probe expression data.

Usage

```
cluster_analysis(sel.exp, cluster_type = "HClust", seed = NULL,
  distance = "euclidean", linkage_type = "ward.D2",
  gene_distance = "correlation", num_clusters, data_name,
  probe_rank = "SD_Rank", probe_num_selection = "Fixed_Probe_Num",
  cluster_num_selection = "Fixed_Clust_Num")
```

Arguments

`sel.exp`	Object containing the numeric selected gene expression matrix. This object is an output of the probe_ranking function.
`cluster_type`	String indicating the type of clustering method to use. "Kmeans" or "HClust" are the two options. The default is set to "HClust".
`seed`	A positive integer vector >1 indicating a random starting position for the centers of the clusters to be used for the k-means clustering algorithm. The default value is set to NULL and is not used when Hierarchical clustering is chosen.
`distance`	String describing the distance metric to use for the dist function during hierarchical clustering. dist uses a default distance metric of Euclidean distance. Options include one of "euclidean", "maximum", manhattan", "canberra", "binary", or "minkowski". Kmeans clustering does not use a distance metric. The default value is set to "euclidean".
`linkage_type`	String describing the linkage metric to be used for HClust. The default is set to "ward.D2", however other options include "average", "complete", "median", "centroid", "single", and "mcquitty".
`gene_distance`	String describing the distance measure to be used for the Dist function when performing hierarchical clustering of genes. Options include one of "euclidean",

"maximum", "manhattan", "canberra", "binary", "pearson", "abspearson", "correlation", "abscorrelation", "spearman" or "kendall". The default of gene_distance is set to "correlation". The deafult value is set to "correlation". The argument can be set to NULL when Kmeans clustering is used.

`num_clusters` Positive integer to specify the number of clusters samples will be divided into. This number is determined by the number_clusters function.

`data_name` String indicating the cancer type and name of the dataset being analyzed. This name will be used to label the sample dendrograms and heatmap files.

`probe_rank` String indicating the feature selection method used in the probe_ranking function. Options include "CV_Rank", "CV_Guided", "SD_Rank", and "Poly".

`probe_num_selection`
String indicating the way in which probes were selected in the number_probes function. Options include "Fixed_Probe_Num", "Percent_Probe_Num", and "Adaptive_Probe_Num".

`cluster_num_selection`
String indicating how the number of clusters were determined in the number_clusters function. Options include "Fixed_Clust_Num" and "Gap_Statistic".

Value

Returns a vector containing the sample information and respective cluster number. In addition, this function outpus sample cluster dendrogams, average expression for each probe in each cluster, and heatmap images and Java TreeView files for HClust dendrograms.

Author(s)

Nathan Lawlor, Alec Fabbri

See Also

probe_ranking, number_clusters, number_probes, hclust, kmeans, dist

Examples

```
# Example 1: HClust Analysis
# Load in a data file
data_file <- system.file("extdata", "GSE2034.normalized.expression.txt",
    package="multiClust")
data <- input_file(input=data_file)
# Choose 300 genes to select for
gene_num <- number_probes(input=data_file, data.exp=data, Fixed=300,
    Percent=NULL, Adaptive=NULL)
# Choose the "CV_Rank" Method for gene ranking
sel.data <- probe_ranking(input=data_file, probe_number=300,
    probe_num_selection="Fixed_Probe_Num", data.exp=data, method="CV_Rank")
# Choose a fixed cluster number of 3
clust_num <- number_clusters(data.exp=data, Fixed=3, gap_statistic=NULL)

# Call function using HClust parameters
```

```
hclust_analysis <- cluster_analysis(sel.exp=sel.data, cluster_type="HClust",
    seed = NULL, distance="euclidean", linkage_type="ward.D2",
    gene_distance="correlation", num_clusters=3,
    data_name="GSE2034 Breast", probe_rank="CV_Rank",
    probe_num_selection="Fixed_Probe_Num",
    cluster_num_selection="Fixed_Clust_Num")

# Example 2: Kmeans Analysis
# Call function for Kmeans parameters
kmeans_analysis <- cluster_analysis(sel.exp=sel.data, cluster_type="Kmeans",
    seed = 1, distance=NULL, linkage_type=NULL, gene_distance=NULL,
    num_clusters=3, data_name="GSE2034 Breast",
    probe_rank="CV_Rank", probe_num_selection="Fixed_Probe_Num",
    cluster_num_selection="Fixed_Clust_Num")
```

`input_file` *Function to read-in the gene expression file and assign gene probe names as the rownames.*

Description

Function to read-in the gene expression file and assign gene probe names as the rownames.

Usage

```
input_file(input)
```

Arguments

`input`	String indicating the name of the text file containing the gene expression matrix to be read in. This matrix file should have the gene probes in the first column of the matrix. The gene probes will be assigned as the rownames of the matrix.

Value

Returns an object containing the gene expression matrix with the gene probe names as the rownames.

Note

This function works best when using gene expression datasets from Gene Expression Omnibus.

Author(s)

Nathan Lawlor

See Also

`read.table`

Examples

```
# Load in a test file
data_file <- system.file("extdata", "GSE2034.normalized.expression.txt",
    package="multiClust")
data <- input_file(input=data_file)
# View matrix with gene probes assigned as rownames
data[1:4, 1:4]
```

`nor.min.max` *Function to normalize data to bring values into alignment. This function uses feature scaling to normalize values in a dataset between 0 and 1.*

Description

Function to normalize data to bring values into alignment. This function uses feature scaling to normalize values in a dataset between 0 and 1.

Usage

```
nor.min.max(x)
```

Arguments

x	An integer object of numeric value

Value

Returns a numeric value normalized between 0 and 1.

Author(s)

Peiyong Guan

Examples

```
# Load sample dataset
data(iris)
# View sample matrix
iris[1:5, 1:5]
# Coerce sample matrix to numeric values
iris <- t(apply(iris[, 1:4], 1, as.numeric))
#Normalize values in the matrix using the function
data.min.max <- t(apply(iris, 1, nor.min.max))
```

number_clusters *Function to determine the number of clusters to be used to cluster gene probes and samples.*

Description

Function to determine the number of clusters to be used to cluster gene probes and samples.

Usage

```
number_clusters(data.exp, Fixed = 3, gap_statistic = NULL)
```

Arguments

data.exp	The numeric original gene expression matrix to be used for clustering of genes and samples. This object is an output of the input_file function.
Fixed	A positive integer used to represent the number of clusters the samples and probes will be divided into. The default cluster number is set to 3 clusters.
gap_statistic	A logical indicating whether to use the gap_statistic to determine the optimal number of clusters to divide samples into.

Value

An object with the determined number of clusters to use.

Note

The user should only choose either the fixed or gap_statistic option, not both. When using the gap_statistic option, change the argument to TRUE and "Fixed" to NULL.

Author(s)

Nathan Lawlor, Alec Fabbri

See Also

clusGap, probe_ranking

Examples

```
#Example 1: Using a fixed cluster number
# Load in a test file
data_file <- system.file("extdata", "GSE2034.normalized.expression.txt",
package="multiClust")
data <- input_file(data_file)
clust_num <- number_clusters(data.exp=data, Fixed=3, gap_statistic=NULL)

## Not run:
```

```
# Example 2: Using the gap_statistic to determine the optimal cluster number
# Computation time is somewhat long
clust_num <- number_clusters(data.exp=data, Fixed=NULL, gap_statistic=TRUE)

## End(Not run)
```

`number_probes` *Function to determine the number of gene probes to select for in the gene feature selection process.*

Description

Function to determine the number of gene probes to select for in the gene feature selection process.

Usage

```
number_probes(input, data.exp, Fixed = 1000, Percent = NULL, Poly = NULL,
  Adaptive = NULL, cutoff = NULL)
```

Arguments

`input`	String indicating the name of the file containing your gene expression matrix.
`data.exp`	The object containing your numeric gene expression matrix. This matrix is an output of the input_file function previously introduced in this package.
`Fixed`	A positive integer specifying a desired number of gene probes to select for. The default is set to 1000 gene probes.
`Percent`	A positive integer between 0 and 100 indicating the percentage of total gene probes to select for from the dataset.
`Poly`	When TRUE, a mean and variance polynomial method is used to determine the number of gene probes to select for. This method uses three second order polynomials to select for the genes with the most variable mean and standard deviations.
`Adaptive`	When TRUE, Gaussian mixture modeling is used to determine the number of gene probes to select.
`cutoff`	Positive number between 0 and 1 specifying the false discovery rate (FDR) cutoff to use with the Adaptive Gaussian mixture modeling method. The default value is set to NULL. However, when Adaptive is TRUE, cutoff should be a positive integer between 0 and 1. Common values to use are 0.05 or 0.01.

Value

Returns an object with the number of gene probes that will be selected in the gene feature selection process. If the Adaptive option is chosen, Gaussian mixture modeling files containing information about the data's mean, variance, mixing proportion, and gaussian assignment are also outputted.

Note

When using this function, the user should only use one option (Fixed, Percent, Adaptive) at a time. When using one method, all other options should be set to NULL.

This function is not needed to determine the number of gene probes to select for in the Poly gene selection method. The particular Poly method does not use a gene probe number input.

Author(s)

Peiyong Guan, Nathan Lawlor

See Also

input_file

Examples

```
# Example 1: Choosing a fixed gene probe number
# Load in a test file
data_file <- system.file("extdata", "GSE2034.normalized.expression.txt",
    package="multiClust")
data <- input_file(input=data_file)
gene_num <- number_probes(input=data_file, data.exp=data, Fixed=300,
    Percent=NULL, Poly=NULL, Adaptive=NULL, cutoff=NULL)

# Example 2: Choosing 50% of the total selected gene probes in a dataset
gene_num <- number_probes(input=data_file, data.exp=data, Fixed=NULL,
    Percent=50, Poly=NULL, Adaptive=NULL, cutoff=NULL)

# Example 3: Choosing the Poly method
gene_num <- number_probes(input=data_file, data.exp=data, Fixed=NULL,
    Percent=NULL, Poly=TRUE, Adaptive=NULL, cutoff=NULL)
## Not run:
# Example 4: Choosing the Adaptive Gaussian Mixture Modeling method
# Very long computation time, so example will not be run
gene_num <- number_probes(input=data_file, data.exp=data, Fixed=NULL,
    Percent=NULL, Poly=NULL, Adaptive=TRUE, cutoff=0.01)

## End(Not run)
```

---

probe_ranking *Function to select for genes using one of the available gene probe ranking options.*

Description

Function to select for genes using one of the available gene probe ranking options.

Usage

```
probe_ranking(input, probe_number, probe_num_selection = "Fixed_Probe_Num",
  data.exp, method = "SD_Rank")
```

Arguments

input	String indicating the name of the text file containing the gene expression matrix.
probe_number	Positive integer indicating the number of gene probes to be selected as determined by the number_probes function.
probe_num_selection	String indicating the way in which number of probes were selected for. Options include "Fixed_Probe_Num", "Percent_Probe_Num", and "Adaptive_Probe_Num".
data.exp	The object containing the original gene expression matrix. This matrix is outputted by the input_file function.
method	A string indicating the gene probe ranking method to use. Possible options include "CV_Rank", "CV_Guided", "SD_Rank", and "Poly". The default is set to "SD_Rank".

Value

An object containing the selected gene expression matrix for a particular ranking method. In addition a text file containing the selected gene expression data is produced.

Note

CV_Rank is a gene probe ranking method that selects for probes with the highest coefficient of variation within the dataset. CV_Guided is a method that also uses the coefficient of variation of the dataset to select for gene probes. Every probe within the set is then plotted on a mean and standard deviation graph (with SD being the y-axis). A line is plotted starting from the origin with a slope of the coefficient of variation. The mean and standard deviation cutoff moves along this line until an equal or less then number of desired probes is above the cutoff. SD_Rank is a gene probe ranking method that selects for probes with the highest standard deviation within the dataset. Poly is a ranking method that fits three second degree polynomial functions of mean and standard deviation to the dataset to select the most variable probes in the dataset.

Author(s)

Peiyong Guan, Alec Fabbri, Nathan Lawlor

See Also

number_probes, input_file

Examples

```
# Producing a selected gene expression matrix using one of the
   # probe ranking options
# Load in a test file
```

```
data_file <- system.file("extdata", "GSE2034.normalized.expression.txt",
    package="multiClust")
data <- input_file(data_file)
selected_probes <- probe_ranking(input=data_file, probe_number=300,
   probe_num_selection="Fixed_Probe_Num", data.exp=data, method="CV_Rank")
```

`surv_analysis` *Function to produce Kaplan-Meier Survival Plots of selected gene expression data.*

Description

Function to produce Kaplan-Meier Survival Plots of selected gene expression data.

Usage

```
surv_analysis(samp_cluster, clinical, survival_type = "RFS", data_name,
  cluster_type = "HClust", distance = "euclidean",
  linkage_type = "ward.D2", probe_rank = "SD_Rank",
  probe_num_selection = "Fixed_Probe_Num",
  cluster_num_selection = "Fixed_Clust_Num")
```

Arguments

`samp_cluster`	Object vector containing the samples and the cluster number they belong to. This object is an output of the cluster_analysis function.
`clinical`	String indicating the name of the text file containing patient clinical information. The file should be a data frame consisting of two columns. The first column contains the patient survival time information in months. The second column indicates occurrence of a censorship (0) or an event (1).
`survival_type`	String specifying the type of survival event being analyzed. Examples include "Disease-free survival (DFS)", "Overall Survival (OS)", "Relapse-free survival (RFS)", etc.
`data_name`	String indicating the name to be used to label the plot.
`cluster_type`	String indicating the type of clustering method used in the cluster_analysis function. "Kmeans" or "HClust" are the two options.
`distance`	String describing the distance metric uses for HClust in the cluster_analysis function. Options include one of "euclidean", "maximum", manhattan", "canberra", "binary", or "minkowski".
`linkage_type`	String describing the linkage metric use in the cluster_analysis function. Options include "ward.D2", "average", "complete", "median", "centroid", "single", and "mcquitty".
`probe_rank`	String indicating the feature selection method used in the probe_ranking function. Options include "CV_Rank", "CV_Guided", "SD_Rank", and "Poly".

probe_num_selection
: String indicating the way in which probes were selected in the number_probes function. Options include "Fixed_Probe_Num", "Percent_Probe_Num", and "Adaptive_Probee_Num".

cluster_num_selection
: String indicating how the number of clusters were determined in the number_clusters function. Options include "Fixed_Clust_Num" and "Gap_Statistic".

Value

Produces a pdf image of a Kaplan-Meier Survival Plot with Cox Survival P Value. Also returns an object containing the cox survival P value.

Author(s)

Alec Fabbri, Nathan Lawlor

See Also

number_clusters, number_probes, probe_ranking, cluster_analysis, coxph

Examples

```
# Load in a data file
data_file <- system.file("extdata", "GSE2034.normalized.expression.txt",
    package="multiClust")
data <- input_file(input=data_file)
# Choose 300 genes to select for
gene_num <- number_probes(input=data_file, data.exp=data, Fixed=300,
    Percent=NULL, Adaptive=NULL)
# Choose the "CV_Rank" Method for gene ranking
sel.data <- probe_ranking(input=data_file, probe_number=300,
    probe_num_selection="Fixed_Probe_Num", data.exp=data, method="CV_Rank")
# Choose a fixed cluster number of 3
clust_num <- number_clusters(data.exp=data, Fixed=3, gap_statistic=NULL)
# Call function for Kmeans parameters
kmeans_analysis <- cluster_analysis(sel.exp=sel.data, cluster_type="Kmeans",
    distance=NULL, linkage_type=NULL, gene_distance=NULL,
    num_clusters=3, data_name="GSE2034 Breast",
    probe_rank="CV_Rank", probe_num_selection="Fixed_Probe_Num",
    cluster_num_selection="Fixed_Clust_Num")
# Load the clinical outcome file
clin_file <- system.file("extdata", "GSE2034-RFS-clinical-outcome.txt",
    package="multiClust")
# Example of Calling surv_analysis function
surv <- surv_analysis(samp_cluster=kmeans_analysis, clinical=clin_file,
    survival_type="RFS", data_name="GSE2034 Breast", cluster_type="Kmeans",
     distance=NULL, linkage_type=NULL, probe_rank="CV_Rank",
     probe_num_selection="Fixed_Probe_Num",
     cluster_num_selection="Fixed_Cluster_Num")
```

`WriteMatrixToFile` *Function to write a data matrix to a text file.*

Description

Function to write a data matrix to a text file.

Usage

```
WriteMatrixToFile(tmpMatrix, tmpFileName, blnRowNames, blnColNames)
```

Arguments

`tmpMatrix`	The object matrix containing data.
`tmpFileName`	The string name of the text file to write the matrix to.
`blnRowNames`	Logical value indicating if row names of the matrix should be written along with the matrix.
`blnColNames`	Logical value indicating if the column names of the matrix should be written with the matrix.

Value

Text file containing the data matrix.

Author(s)

Peiyong Guan

See Also

`write.table`

Examples

```
#Load sample dataset
data(iris)
# View sample matrix
iris[1:4, 1:4]
# Write sample matrix to text file
WriteMatrixToFile(tmpMatrix=iris, tmpFileName="iris.sample.matrix.txt",
blnRowNames=TRUE, blnColNames=TRUE)
```

Index