Package 'EMVC'

July 2, 2014

Type Package

Title Entropy Minimization over Variable Clusters (EMVC)

Version 0.1

Date 2013-08-13

Author H. Robert Frost and Jason H. Moore

Maintainer H. Robert Frost <rob.frost@dartmouth.edu>

Description Contains logic for the data-driven optimization of annotations via minimization of the entropy of variable group members over discrete variable clusters.

NeedsCompilation no

Repository CRAN

Date/Publication 2013-08-27 18:33:47

R topics documented:

EMVC-package *Implementation of the Entropy Minimization over Variable Clusters (EMVC) algorithm*

Description

Contains logic for the data-driven optimization of annotations via minimization of the entropy of variable group members over discrete partitions of variables generated via either k-means clustering or horizontal cuts of dendrograms output via agglomerative hierarchical clustering.

Details

Package:	EMVC
Type:	Package
Version:	1.0
Date:	2013-08-13
License:	GPL-3

Variable group optimization is performed using the function EMVC.

Note

This work was supported by the National Institutes of Health R01 LM010098.

Author(s)

H. Robert Frost and Jason H. Moore

References

Frost, HR, Moore, JH. Optimization of gene sets via entropy minimization over variable clusters (EMVC). Submitted.

EMVC *Entropy Minimization over Variable Clusters (EMVC) algorithm*

Description

Implementation of the EMVC algorithm. Takes an n-by-p data matrix and a c-by-p binary annotation matrix and generates an optimized, i.e., filtered, version of the annotation matrix by minimizing the entropy between each variable group and the categorical random variable representing membership of each variable in clusters output by either k-means clustering or horizontal cuts of a dendrogram generated via agglomerative hierarchical clustering with correlation distance. Annotations are never added during optimization, just removed.

Usage

```
EMVC(data, annotations, bootstrap.iter=20, k.range=NA, clust.method="kmeans",
     kmeans.nstart=1, kmeans.iter.max=10, hclust.method="average",
     hclust.cor.method="spearman")
```

Arguments

`data`	Input data matrix, observations-by-variables. Must be specified. Cannot contain missing values.
`annotations`	Binary annotation matrix, variable groups-by-variables. Must be specified.
`bootstrap.iter`	Number of bootstrap iterations. Defaults to 20. If set to 1, will return the results from a single optimization run on the input data matrix (i.e., no bootstrapping will be performed).
`clust.method`	Method used to generate variable clusters. Either "kmeans" or "hclust". Defaults to "kmeans".
`k.range`	Range of k-means k values or dendrogram cut sizes. Must be specified.
`kmeans.nstart`	Only relevant if clust.method is "kmeans". K-means nstart value. Defaults to 5.
`kmeans.iter.max`	Only relevant if clust.method is "kmeans".Max number of iterations for k-means. Defaults to 20.
`hclust.method`	Only relevant if clust.method is "hclust". Will be supplied as the "method" argument to the R function `hclust`. Defaults to "average".
`hclust.cor.method`	Only relevant if clust.method is "hclust". Will be supplied as the "method" argument to the R `cor` function. Defaults to "spearman". Represents the correlation method used to compute the dissimilarity matrix for `hclust`. Entries in the dissimilarity matrix will take the form (1-correlation)/2.

Value

Optimized version of the annotation matrix. Contains the average proportion of cluster sizes in which a given annotation was kept during optimization. If bootstrapping is enabled, the optimized matrix will contain the average proportions over all bootstrap resampled datasets.

See Also

`filterAnnotations`.

Examples

```
## Create random sparse annotation matrix for 50 variable groups
## and 100 variables
annotations = matrix(rbinom(5000,1,.1), nrow=50, ncol=100)

## Number of initial annotations
sum(annotations)
```

```
## Create random gene expression matrix for 50 observations and 100 variables
data = matrix(rnorm(5000), nrow=50, ncol=100)

## Execute EMVC using k-means
EMVC.results = EMVC(data=data, annotations=annotations,
                    bootstrap.iter=30, k.range=2:10, clust.method="kmeans",
                    kmeans.nstart=3, kmeans.iter.max=10)

## Filter the results at .9 threshold
filtered.opt.annotations = filterAnnotations(EMVC.results, .9)

## Number of optimized annotations at .9 threshold, should be close to 0 since the
## variable groups and data are random (i.e., no random annotations avoid
## optimization-based filtering most of the time)
sum(filtered.opt.annotations)

## Filter the results at .1 threshold
filtered.opt.annotations = filterAnnotations(EMVC.results, .1)

## Number of optimized annotations at .1 threshold, should be close to
## the initial number of annotations since the variable groups and data are random
## (i.e., no random variables are consistently filtered by the EMVC algorithm)
sum(filtered.opt.annotations)
```

`filterAnnotations` *Filter output of EMVC algorithm*

Description

Filters the output of the EMVC algorithm to remove all annotations whose bootstrap annotation probability is below a specified threshold.

Usage

```
filterAnnotations(annotations, proportion)
```

Arguments

`annotations`	Optimized annotation matrix returned by function `EMVC`. Rows represent gene sets and columns represent genes. Elements of the matrix represent the proportion of clusterings within all bootstrap resampled data sets in which the annotation was not filtered by EMVC.
`proportion`	Threshold for annotation filtering. All annotations whose optimization proportion is below this value will be filtered.

Value

Updated version of the input annotation matrix that contains only those annotations with proportions greater than or equal to the specified threshold. Elements in this matrix are either 0 or 1.

See Also

EMVC.

Examples

```
## Create random optimized annotation matrix for 50 gene sets and 100 genes.
## This mimics what is generated by the EMVC() function
opt.annotations = matrix(runif(5000), nrow=50, ncol=100)

## Total number of non-zero annotations
length(which(opt.annotations > 0))

## Filter the results at .5 threshold. The number should be approximately half
## of the non-zero annotations since the proportions were generated as uniform(0,1)
filtered.opt.annotations = filterAnnotations(opt.annotations, .5)
sum(filtered.opt.annotations)
```

Index