

# MantelCorr

April 19, 2010

---

ClusterGeneList      *Generate Genes from a Cluster List*

---

## Description

'ClusterGeneList' produces a list of both significant and nonsignificant genes from each respective cluster type

## Usage

```
ClusterGeneList(clus, clustlist.sig, x.data)
```

## Arguments

clus	'clusters' object returned by 'GetClusters'
clustlist.sig	'SignificantClusters' object returned by 'ClusterList'
x.data	original (p x n) numeric data matrix (e.g., gene-expression data)

## Value

A list with components:

SignificantClusterGenes	significant cluster genes returned from 'ClusterList'
NonSignificantClusterGenes	nonsignificant cluster genes returned from 'ClusterList'

## Note

argument 'x.data' should have an ID gene variable, 'probes', attached as a 'dimnames' attribute

## Author(s)

Brian Steinmeyer

## See Also

'GetClusters' 'ClusterList'

**Examples**

```
# simulate a p x n microarray expression dataset, where p = genes and n = samples
data.sep <- rbind(matrix(rnorm(1000), ncol=50), matrix(rnorm(1000, mean=5), ncol=50))
noise <- matrix(runif(40000), ncol=1000)
data <- t(cbind(data.sep, noise))
data <- data[1:200, ]
# data has p = 1,050 genes and n = 40 samples

clusters.result <- GetClusters(data, 100, 100)
dist.matrices <- DistMatrices(data, clusters.result$clusters)
mantel.corr <- MantelCorrs(dist.matrices$Dfull, dist.matrices$Dsubsets)
permutation.result <- PermutationTest(dist.matrices$Dfull, dist.matrices$Dsubsets, 100, 4)

# generate both significant and non-significant gene clusters
cluster.list <- ClusterList(permutation.result, clusters.result$cluster.sizes, mantel.corr)

# significant and non-significant cluster genes (expression values)
cluster.genes <- ClusterGeneList(clusters.result$clusters, cluster.list$SignificantClusters)
```

ClusterList

*Generate a Cluster List***Description**

'ClusterList' generates a list of both significant and nonsignificant clusters, with cluster number, Mantel cluster correlation and size

**Usage**

```
ClusterList(p.val, clus.size, mantel.corr)
```

**Arguments**

`p.val` permutation p-value returned from 'PermutationTest'  
`clus.size` vector of k cluster sizes returned from 'GetCluster'  
`mantel.corr` original, unpermuted k Mantel correlations returned from 'MantelCorrs'

**Value**

A list with components:

`SignificantClusters`  
clusters with significant Mantel correlation, equal to or larger than the permutation p-value returned by 'PermutationTest'  
`NonSignificantClusters`  
clusters with nonsignificant Mantel correlation, smaller than the permutation p-value returned by 'PermutationTest'

**Author(s)**

Brian Steinmeyer

**See Also**

'PermutationTest'

**Examples**

```
# simulate a p x n microarray expression dataset, where p = genes and n = samples
data.sep <- rbind(matrix(rnorm(1000), ncol=50), matrix(rnorm(1000, mean=5), ncol=50))
noise <- matrix(runif(40000), ncol=1000)
data <- t(cbind(data.sep, noise))
data <- data[1:200, ]
# data has p = 1,050 genes and n = 40 samples

clusters.result <- GetClusters(data, 100, 100)
dist.matrices <- DistMatrices(data, clusters.result$clusters)
mantel.corrs <- MantelCorrs(dist.matrices$Dfull, dist.matrices$Dsubsets)
permutation.result <- PermutationTest(dist.matrices$Dfull, dist.matrices$Dsubsets, 100, 4)

# generate both significant and non-significant gene clusters
cluster.list <- ClusterList(permutation.result, clusters.result$cluster.sizes, mantel.corrs)
```

---

DistMatrices

*Compute Dissimilarity Matrices*


---

**Description**

'DistMatrices' uses 'dist' to compute dissimilarity matrices for 'data' and each cluster k from 'GetClusters'

**Usage**

```
DistMatrices(x.data, cluster.assignment)
```

**Arguments**

x.data            original 'data' matrix  
cluster.assignment    cluster assignment vector, "clusters", returned by 'GetClusters'

**Value**

returns a list with two components:

Dsubsets        dissimilarity matrices for each cluster k  
Dfull            dissimilarity matrix for the original 'data'

**Note**

'GetClusters' should be executed prior to 'DistMatrices'

**Author(s)**

Brian Steinmeyer

**See Also**

'GetClusters'

**Examples**

```
# simulate a p x n microarray expression dataset, where p = genes and n = samples
data.sep <- rbind(matrix(rnorm(1000), ncol=50), matrix(rnorm(1000, mean=5), ncol=50))
noise <- matrix(runif(40000), ncol=1000)
data <- t(cbind(data.sep, noise))
data <- data[1:200, ]
# data has p = 1,050 genes and n = 40 samples

clusters.result <- GetClusters(data, 100, 100)
dissimilarity.matrices <- DistMatrices(data, clusters.result$clusters)
```

---

GetClusters

*Over-Partition a (p x n) Data Matrix using 'kmeans'*

---

**Description**

'GetClusters' uses an overly large k with the 'kmeans' function to over-partition p variables (rows = genes) from n objects (cols = samples) from a given data matrix 'x.data'

**Usage**

```
GetClusters(x.data, num.k, num.iters)
```

**Arguments**

x.data	p x n data matrix of numeric values
num.k	number of k partitions desired
num.iters	number of iterations - recommend >= 100

**Value**

'GetClusters' returns a list with the following components:

clusters	cluster assignment from 'kmeans'
cluster.sizes	size of each cluster k from 'kmeans'

**Note**

The input data matrix, x.data, must be numeric (e.g., gene-expression values). We recommend using 'num.k' = one-half the number of genes and 'num.iters' greater than 50

**Author(s)**

Brian Steinmeyer

**See Also**

'kmeans'

**Examples**

```
# simulate a p x n microarray expression dataset, where p = genes and n = samples
data.sep <- rbind(matrix(rnorm(1000), ncol=50), matrix(rnorm(1000, mean=5), ncol=50))
noise <- matrix(runif(40000), ncol=1000)
data <- t(cbind(data.sep, noise))
data <- data[1:200, ]
# data has p = 1,050 genes and n = 40 samples

clusters.result <- GetClusters(data, 100, 100)
```

---

GolubTrain

*Golub Training Set*

---

**Description**

Samples were taken with Affymetrix Hgu6800 chips and expression levels measured on 7,129 genes (probes). The samples consist of 27 acute lymphoblastic leukemia (ALL) and 11 acute myeloid leukemia (AML) patients. The data values are raw (e.g. no standardization or gene filtering applied).

**Usage**

```
data(GolubTrain)
```

**Format**

A data frame of 7129 observations (genes) with the following 38 variables (samples):

**X1** ALL

**X2** ALL

**X3** ALL

**X4** ALL

**X5** ALL

**X6** ALL

**X7** ALL

**X8** ALL

**X9** ALL

**X10** ALL

**X11** ALL

X12 ALL  
X13 ALL  
X14 ALL  
X15 ALL  
X16 ALL  
X17 ALL  
X18 ALL  
X19 ALL  
X20 ALL  
X21 ALL  
X22 ALL  
X23 ALL  
X24 ALL  
X25 ALL  
X26 ALL  
X27 ALL  
X28 AML  
X29 AML  
X30 AML  
X31 AML  
X32 AML  
X33 AML  
X34 AML  
X35 AML  
X36 AML  
X37 AML  
X38 AML

**Source**

<http://www.broad.mit.edu/cgi-bin/cancer/datasets.cgi>

**References**

Golub, T.R. Molecular Classification of Cancer: Class Discovery and Class Prediction by Gene Expression Monitoring. *Science*, vol 286, 531-537, 1999.

**Examples**

```
data(GolubTrain)
```

---

MantelCorrs	<i>Compute Mantel Correlation(s)</i>
-------------	--------------------------------------

---

**Description**

'MantelCorrs' computes the Mantel correlation between two dissimilarity matrices

**Usage**

```
MantelCorrs(Dfull, Dsubsets)
```

**Arguments**

Dfull	distance matrix returned by 'DistMatrices' using original 'data'
Dsubsets	list of distance matrices from each k cluster or partition returned by 'DistMatrices'

**Value**

A list with k components

where component i  
Mantel correlation for cluster i,  $i = 1, \dots, k$

**Warning**

The function is meant to be executed AFTER 'GetClustes' and 'DistMatrices' (see example)

**Note**

the value 'k' corresponds to the parameter 'num.k' in 'GetClusters'

**Author(s)**

Brian Steinmeyer

**References**

Mantel N: The detection of disease clustering and a generalized regression approach. *Cancer Research*. 27(2), 209-220 (1967).

**See Also**

'GetClusters' 'DistMatrices' 'kmeans'

**Examples**

```
# simulate a p x n microarray expression dataset, where p = genes and n = samples
data.sep <- rbind(matrix(rnorm(1000), ncol=50), matrix(rnorm(1000, mean=5), ncol=50))
noise <- matrix(runif(40000), ncol=1000)
data <- t(cbind(data.sep, noise))
data <- data[1:200, ]
# data has p = 1,050 genes and n = 40 samples

clusters.result <- GetClusters(data, 100, 100)
dist.matrices <- DistMatrices(data, clusters.result$clusters)
mantel.corr <- MantelCorrs(dist.matrices$Dfull, dist.matrices$Dsubsets)
```

---

PermutatonTest      *Permutation Test for Dissimilarity Matrices*

---

**Description**

'PermutationTest' computes and returns an empirical p-value from a null distribution generated by permuting 'Dfull' a total of 'num.per' times.

**Usage**

```
PermutationTest(Dfull, Dsubsets, num.per, num.chips, alpha)
```

**Arguments**

Dfull	dissimilarity matrix from the original (p x n) microarray expression data
Dsubsets	dissimilarity matrices from each k disjoint clusters returned by 'GetClusters'
num.per	number of permutations
num.chips	number of samples, 'n' from the original (p x n) data matrix
alpha	desired level of significance

**Details**

For each permutation, k Mantel correlations are computed by correlating the permuted 'Dfull' with each dissimilarity matrix 'Dsubsets' from the 'k' clusters returned by 'GetClusters'. The absolute value of the maximum Mantel cluster correlation is retained at each permutation. These 'num.per' maximum correlations are then used to generate a null distribution for distance metric independence, with the p-value taken from the (1 - 'alpha') percentile of this permutation distribution.

**Value**

returns the permuted p-value for the 'alpha' selected level of significance

**Warning**

(p x n) data matrix should be numeric (e.g. gene-expression levels)



**Note**

The function is meant to be executed AFTER 'GetClustes', 'DistMatrices' and 'MantelCorr' (see example)

**Author(s)**

Brian Steinmeyer

**See Also**

'GetClusters' 'DistMatrices' 'MantelCorrs'

**Examples**

```
# simulate a p x n microarray expression dataset, where p = genes and n = samples
data.sep <- rbind(matrix(rnorm(1000), ncol=50), matrix(rnorm(1000, mean=5), ncol=50))
noise <- matrix(runif(40000), ncol=1000)
data <- t(cbind(data.sep, noise))
data <- data[1:200, ]
# data has p = 1,050 genes and n = 40 samples

clusters.result <- GetClusters(data, 100, 100)
dist.matrices <- DistMatrices(data, clusters.result$clusters)
mantel.corrs <- MantelCorrs(dist.matrices$Dfull, dist.matrices$Dsubsets)
permutation.result <- PermutationTest(dist.matrices$Dfull, dist.matrices$Dsubsets, 100, 4
```

# Index

## \*Topic **cluster**

ClusterGeneList, 1

ClusterList, 2

DistMatrices, 3

GetClusters, 4

MantelCorrs, 7

PermutatonTest, 8

## \*Topic **datasets**

GolubTrain, 5

ClusterGeneList, 1

ClusterList, 2

DistMatrices, 3

GetClusters, 4

GolubTrain, 5

MantelCorrs, 7

PermutationTest (*PermutatonTest*),  
8

PermutatonTest, 8