

# Package: MMOC (via r-universe)

September 4, 2024

**Title** Multi-Omic Spectral Clustering using the Flag Manifold

**Version** 0.1.1.0

**Maintainer** Charlie Carpenter <charles.carpenter@cuanschutz.edu>

**Description** Multi-omic (or any multi-view) spectral clustering methods often assume the same number of clusters across all datasets. We supply methods for multi-omic spectral clustering when the number of distinct clusters differs among the omics profiles (views).

**License** MIT + file LICENSE

**Encoding** UTF-8

**RoxygenNote** 7.2.1

**Imports** Spectrum (>= 1.1), igraph (>= 1.4.1), MASS (>= 7.3-58.1)

**Suggests** knitr, rmarkdown, testthat (>= 3.0.0), SNFtool (>= 2.3.1), plotly (>= 4.10.0)

**Config/testthat/edition** 3

**VignetteBuilder** knitr

**NeedsCompilation** no

**Author** Charlie Carpenter [aut, cre]  
(<<https://orcid.org/0000-0002-4698-1516>>)

**Date/Publication** 2023-08-10 10:20:09 UTC

**Repository** <https://charliecarpenter.r-universe.dev>

**RemoteUrl** <https://github.com/cran/MMOC>

**RemoteRef** HEAD

**RemoteSha** 561deebd2e8e16df217025885e84d209399aa131

## Contents

clustStruct . . . . .	2
flagMean . . . . .	3
kernelLaplacian . . . . .	4

Laplacian . . . . .	6
Lapprox . . . . .	7
trueGroups . . . . .	8

<b>Index</b>	<b>9</b>
--------------	----------

---

clustStruct	<i>Generate multi-view data sets with simple cluster structures</i>
-------------	---

---

## Description

Generates multiple data sets from a multivariate normal distribution using the [mvrnorm](#) function from the MASS package.

## Usage

```
clustStruct(n, p, k, noiseDat = "random", randNoise = 2)
```

## Arguments

n	An integer, the sample size for all generated data sets
p	An integer, the number of columns (features) in each generated data set
k	An integer or vector, the number of distinct clusters in each generated data set. n/k must be an integer for all values of k
noiseDat	Either the character string 'random', indicating the covariance matrix is a diagonal matrix with randNoise along the diagonal, or a valid covariance matrix
randNoise	The value along the diagonal when noiseDat='random'

## Details

The function accepts k as a vector. It splits data into k groups with means  $c(0, 2^{(1:(k-1))})$ , e.g., when k=3 the data will be split into 3 groups with means 0, 2, and 4, respectively. The covariance matrix is either a diagonal matrix with randNoise (an integer) along the diagonal, or a given matrix.

## Value

A list of  $n \times p$  data frames with the specified number of groups

## Examples

```
## A single view with 30 variables and 3 groups
s1 <- clustStruct(n=120, p=30, k=3, noiseDat='random')[[1]]

## Multiple views with 30 variables
## View 1 has 2 groups and View 2 has 3 groups
s2 <- clustStruct(n=120, p=30, k=c(2,3), noiseDat='random')
```

```
## Multiple views with 30 variables
## View 1 has 2 groups, View 2 has 3, and View 3 has 3 groups
s3 <- clustStruct(n=120, p=30, k=c(2,3,3), noiseDat='random')

## Three view study.
# View 1: 2 groups, 30 variables, random noise = 5
# View 2: 3 groups, 60 variables, random noise = 2
# View 3: 4 groups, 45 variables, random noise = 4

s4 <- clustStruct(n=120, k=c(2,3,4), p=c(30,60,45), randNoise=c(5,2,4))
```

---

flagMean

*Calculate the Flag mean of multiple subspaces*


---

### Description

Calculate the flag-mean of multiple subspaces. This method allows you to find the extrinsic mean of a finite set of subspaces. You can think of this as a median subspace. This method is also able to handle subspaces with different dimensions. See the references for more details

### Usage

```
flagMean(LapList, k, laplacian = c("shift", "Ng", "sym", "rw"), plots = TRUE)
```

### Arguments

LapList	A list of Laplacian matrices
k	A vector indicating how many eigenvectors to take from each Laplacian, i.e., the number of clusters in each view
laplacian	One of "shift", "Ng", "rw" or "sym". Should be the same type used to calculate your Laplacians
plots	Whether or not to plot the singular values from SVD

### Details

Despite the complex linear algebra to achieve this result, the operation is very simple. This function concatenates (cbind) the given subspaces and then performs singular value decomposition on the resulting matrix. This gives the 'median' subspace of the given set of subspaces. We would then cluster on the columns of the U matrix just as we do in standard spectral clustering

### Value

The output from a singular value decomposition. See [svd](#)

### References

<https://www.semanticscholar.org/paper/Flag-Manifolds-for-the-Characterization-of-in-Large-Marrinan-Beveridge/7d306512b545df98243f87cb8173df83b4672b18> <https://www.sciencedirect.com/science/article/pii/S0024379514>

**Examples**

```

## Generating data with 2 and 3 distinct clusters
## Note that 'clustStruct' returns a list
n=120; k <- c(2,3)
dd <- clustStruct(n=n, p=30, k=k, noiseDat='random')

## Laplacians
L_list <- lapply(dd, kernelLaplacian, kernel="Spectrum", plots=FALSE, verbose=FALSE)

## Calculating the flag mean
fm <- flagMean(L_list, k=k, laplacian='shift')

## Knowing the true structure makes it much easier to know how
## many right singular vectors to grab. There are 4 distinct
## groups in these data from 'clustStruct'

trueGroups(n=n, k=k)

kmeans(fm$u[, 1:4], centers=4)

```

---

kernelLaplacian

*Calculate the graph Laplacian of a given data set*


---

**Description**

Calculate the graph laplacian from a given data set with subjects as rows and features as columns.

**Usage**

```

kernelLaplacian(
  dat,
  kernel = c("Gaussian", "ZM", "Spectrum", "Linear"),
  laplacian = c("shift", "Ng", "sym", "rw"),
  grf.type = c("full", "knn", "e-graph"),
  k = 5,
  p = 5,
  rho = NULL,
  epsilon = 0,
  mutual = FALSE,
  binary.grf = FALSE,
  plots = TRUE,
  verbose = TRUE
)

```

**Arguments**

**dat** A matrix like object with subjects as rows and features as columns.

kernel	The type of kernel used to calculate the graph's adjacency matrix: "Gaussian" for the standard Gaussian kernel, "ZM" for the Zelnik-Manor kernel, "Spectrum" for the spectrum kernel, "Linear" for the linear kernel (dot product), and "Cor" for a kernel of pairwise correlations. See references for more details.
laplacian	One of "shift", "Ng", "rw" or "sym". See details for description
grf.type	Type of graph to calculate: "full" for adjacency matrix equal to the kernel, "knn" for a k-nearest neighbors graph, "e-graph" for an "epsilon graph"
k	An integer value for k in the k-nearest neighbors graph. Only the k largest edges (most similar neighbors) will be kept
p	An integer value for the p-nearest neighbor in the ZM kernel
rho	A value for the dispersion parameter in the Gaussian kernel. It is in the denominator of the exponent, so higher values correspond to lower similarity. By default it is the median pairwise Gaussian distance
epsilon	The cutoff value for the e-graph. Edges lower than this value will be removed
mutual	Make a "mutual" knn graph. Only keeps edges when two nodes are both in each others k-nearest set
binary.grf	Set all edges >0 to 1
plots	Whether or not to plot the final graph, a heatmap of calculated kernel, and the eigen values of the Laplacian
verbose	Whether or not to give some summary statistics of the pairwise distances

### Details

The four Laplacians are defined as  $L_{shift} = I + D^{-1/2}AD^{-1/2}$ ,  $L_{Ng} = D^{-1/2}AD^{-1/2}$ ,  $L_{sym} = I - D^{-1/2}AD^{-1/2}$ , and  $L_{rw} = I - D^{-1}A$ . The shifted Laplacian,  $L_{shift} = I + D^{-1/2}AD^{-1/2}$ , is recommended for multi-view spectral clustering.

### Value

An  $n \times n$  matrix where n is the number of rows in dat.

### References

<https://academic.oup.com/bioinformatics/article/36/4/1159/5566508#199177546>

### Examples

```
## Generating data with 3 distinct clusters
## Note that 'clustStruct' returns a list
dd <- clustStruct(n=120, p=30, k=3, noiseDat='random')[[1]]

kernelLaplacian(dd, kernel="Spectrum")
```

---

Laplacian

*Calculate the graph Laplacian from a given adjacency matrix*


---

### Description

Calculate the graph laplacian from a given kernel matrix that represents the full graph weighted adjacency matrix

### Usage

```
Laplacian(
  A,
  laplacian = c("shift", "Ng", "sym", "rw"),
  grf.type = c("full", "knn", "e-graph"),
  k = 5,
  rho = NULL,
  epsilon = 0,
  mutual = FALSE,
  binary.grf = FALSE,
  plots = TRUE
)
```

### Arguments

A	An n by n kernel matrix, where n is the sample size, that represents your initial adjacency matrix. Kernel matrices are symmetric, positive semi-definite distance matrices
laplacian	One of "shift", "Ng", "rw" or "sym". See details for description
grf.type	Type of graph to calculate: "full" for adjacency matrix equal to the kernel, "knn" for a k-nearest neighbors graph, "e-graph" for an "epsilon graph"
k	An integer value for k in the k-nearest neighbors graph. Only the k largest edges (most similar neighbors) will be kept
rho	A value for the dispersion parameter in the Gaussian kernel. It is in the denominator of the exponent, so higher values correspond to lower similarity. By default it is the median pairwise Gaussian distance
epsilon	The cutoff value for the e-graph. Edges lower than this value will be removed
mutual	Make a "mutual" knn graph. Only keeps edges when two nodes are both in each others k-nearest set
binary.grf	Set all edges >0 to 1
plots	Whether or not to plot the final graph, a heatmap of calculated kernel, and the eigen values of the Laplacian

**Details**

The four Laplacians are defined as  $L_{shift} = I + D^{-1/2}AD^{-1/2}$ ,  $L_{Ng} = D^{-1/2}AD^{-1/2}$ ,  $L_{sym} = I - D^{-1/2}AD^{-1/2}$ , and  $L_{rw} = I - D^{-1}A$ . The shifted Laplacian,  $L_{shift} = I + D^{-1/2}AD^{-1/2}$ , is recommended for multi-view spectral clustering.

**Value**

An  $n \times n$  matrix where  $n$  is the number of rows in `dat`.

**References**

<https://academic.oup.com/bioinformatics/article/36/4/1159/5566508#199177546>

**Examples**

```
## Generating data with 3 distinct clusters
## Note that 'clustStruct' returns a list
dd <- clustStruct(n=120, p=30, k=3, noiseDat='random')[[1]]

## Gaussian kernel
rho <- median(dist(dd))
A <- exp(-(1/rho)*as.matrix(dist(dd, method = "euclidean", upper = TRUE)^2))

Laplacian(A, laplacian='shift', grf.type = 'knn')
```

---

Lapprox

---

*Compute a rank k approximation of a graph Laplacian*


---

**Description**

This function calculates the rank-k approximation of a graph Laplacian (or any symmetric matrix). This function performs [eigen](#) decomposition on the given matrix `L` and reconstructs it using only the LAST `k` eigenvectors and eigenvalues.

**Usage**

```
Lapprox(LapList, k, laplacian = c("shift", "Ng", "sym", "rw"), plots = TRUE)
```

**Arguments**

<code>LapList</code>	A list of Laplacian matrices
<code>k</code>	A vector indicating how many eigenvectors to take from each Laplacian, i.e., the number of clusters in each view
<code>laplacian</code>	One of "shift", "Ng", "rw" or "sym". Should be the same type used to calculate your Laplacians
<code>plots</code>	Whether or not to plot the eigenvalues from the rank approximated Laplacians

**Value**

An  $n \times n$  matrix

**Examples**

```
## Generating data with 2 and 3 distinct clusters
## Note that 'clustStruct' returns a list
n=120; k <- c(2,3)
set.seed(23)
dd <- clustStruct(n=n, p=30, k=k, noiseDat='random')

## Laplacians
L_list <- lapply(dd, kernelLaplacian, kernel="Spectrum",
  laplacian='shift', plots=FALSE, verbose=FALSE)

trueGroups(n,k)

La <- Lapprox(L_list, k=k, plots=FALSE)

kmeans(La$vector[,1:4], centers=4)
```

---

trueGroups

*Get the groups created by the clustStruct function*


---

**Description**

Get the unique groups generated by the [clustStruct](#) function for a given  $k$ . The number of rows of the resulting matrix gives the number of unique groups.

**Usage**

```
trueGroups(n, k)
```

**Arguments**

$n$  An integer, the sample size for all generated data sets

$k$  An integer or vector, the number of distinct clusters in each generated data set.  $n/k$  must be an integer for all values of  $k$

**Value**

A matrix with the unique groups/clusters from the multi-view data generated from [clustStruct](#). The final column Grps enumerates these groups.

**Examples**

```
trueGroups(n=120, k=c(2,3,4))
```



# Index

clustStruct, [2](#), [8](#)

eigen, [7](#)

flagMean, [3](#)

kernellaplacian, [4](#)

Laplacian, [6](#)

Lapprox, [7](#)

mvrnorm, [2](#)

svd, [3](#)

trueGroups, [8](#)