

# CARP: The Clustering Algorithms' Referee Package

## Version 3.3 – Manual

Volodymyr Melnykov\*  
Department of Info Systems,  
Statistics, and Management Science  
The University of Alabama  
Tuscaloosa, AL 35487  
USA.

E-mail: [vmelnykov@ua.edu](mailto:vmelnykov@ua.edu)

Ranjan Maitra\*  
Department of Statistics  
Iowa State University  
Ames, IA 50011  
USA.

E-mail: [maitra@iastate.edu](mailto:maitra@iastate.edu)

November 7, 2013

---

\*Supported in part by the National Science Foundation (NSF) through its CAREER Award No. DMS-0437555.

# Contents

<b>1</b>	<b>Description</b>	<b>4</b>
1.1	Project homepage . . . . .	4
1.2	Dependencies . . . . .	4
<b>2</b>	<b>Installation</b>	<b>4</b>
<b>3</b>	<b>Usage</b>	<b>5</b>
3.1	Parameters . . . . .	5
3.2	Details . . . . .	6
3.2.1	Organization of Files . . . . .	6
3.2.2	Arguments and Options for CARP . . . . .	6
3.3	Other capabilities . . . . .	8
3.3.1	Some useful features . . . . .	8
3.3.2	Using CARP standalone to generate mixture models and datasets . .	9
3.3.3	Using external clustering procedures . . . . .	9
3.3.4	Using external metrics for assessing performance of clustering procedures	9
3.3.5	Simulating non-Gaussian mixtures . . . . .	9
3.3.6	Simulating noisy coordinates . . . . .	10
3.3.7	Simulating datasets with outliers . . . . .	10
<b>4</b>	<b>Illustrations</b>	<b>10</b>
4.1	Example: Evaluating clustering algorithms with spherical dispersion struc- tures and equal representation . . . . .	10
4.2	Example: Evaluating clustering algorithms with homogeneous non-spherical dispersion structures and unequal representation . . . . .	14
4.3	Example: Evaluating a clustering algorithm from data simulated earlier and stored in files . . . . .	16
4.4	Example: Calculating overlap between identified classes in a dataset . . . . .	17
4.5	Example: Use of CARP as a standalone package . . . . .	18
4.6	Example: Using external clustering programs - the R function PAM . . . . .	19
4.6.1	Expert Mode for the Advanced User . . . . .	19
4.6.2	Command-Line interface for the Less-Experienced User . . . . .	22
4.6.3	Command-Line Interface: An alternative approach . . . . .	24
4.7	Example: Using external clustering procedures - the R function Mclust . . .	25
4.7.1	Expert Mode for the Advanced User . . . . .	25
4.7.2	Command-Line interface for the Less-Experienced User . . . . .	28
4.7.3	Command-Line Interface: An alternative approach . . . . .	30
4.8	Example: Using external clustering metrics - the R function BHI . . . . .	30
4.8.1	Expert Mode for the Advanced User . . . . .	30
4.8.2	Command-Line interface for the Less-Experienced User . . . . .	31
4.8.3	Command-Line Interface: An alternative approach . . . . .	34
4.9	Example: Using external clustering metrics - Diag using the corrected R function classAgreement . . . . .	34



4.9.1	Expert Interface for the Advanced User . . . . .	34
4.9.2	Command-Line interface for the Less-Experienced User . . . . .	36
4.9.3	Command-Line Interface: An alternative approach . . . . .	39
4.10	Example: Simulating non-Gaussian mixtures . . . . .	39
4.11	Example: Simulating noise coordinates . . . . .	40
4.12	Example: Simulating datasets with outliers . . . . .	43
<b>5</b>	<b>An Illustrative Comprehensive Example: Comparing several clustering algorithms</b>	<b>45</b>
<b>6</b>	<b>Illustration of CARP to evaluate separation on a given dataset: <i>Iris</i></b>	<b>46</b>
<b>7</b>	<b>Conclusions</b>	<b>47</b>
<b>A</b>	<b>Appendix: The R package MIXSIM</b>	<b>47</b>
A.1	MixSim: mixture simulation . . . . .	48
A.2	simdataset: dataset simulation . . . . .	49
A.3	overlap: overlap calculation . . . . .	50
A.4	pdplot: parallel distribution plot . . . . .	52
A.5	RandIndex: index calculation . . . . .	53
A.6	VarInf: variation of information calculation . . . . .	54
A.7	VarInf: variation of information calculation . . . . .	54
<b>B</b>	<b>Appendix: Geometry of Overlap</b>	<b>55</b>

# 1 Description

The C-package CARP provides an easy and convenient integrated tool for evaluating performance of clustering algorithms. The underlying methodology is based on first simulating Gaussian mixture models according to prespecified levels of average, maximum, or generalized pairwise overlaps. The concept of overlap is defined as the sum of two misclassification probabilities (Maitra and Melnykov, 2010a). Datasets are then simulated from the realized Gaussian mixtures. This concludes the first phase of the procedure. In the second phase, the clustering algorithm being evaluated is run on the generated datasets. We provide several examples here using an agglomerative hierarchical clustering algorithm (**hierclust**),  $k$ -means algorithm (**Kmeans**), partitioning around medoids algorithm (**PAM**), and the EM algorithm (**Mclust**). The latter are used directly from related R packages and also provide examples on how to use CARP with R. The third phase compares the derived partitionings with the true. By default, the comparison measure is the adjusted Rand index of Hubert and Arabie (1985) but the user can also provide some other measures in executable form, or a form that can be launched from the command line. Illustrations for the latter are provided in the form of the **BHI**, **Rand**, and **Diag**. Upon conclusion, CARP provides a distribution of the desired performance measure for the clustering method being evaluated at the preferred setting. This provides for a detailed understanding of the performance of the clustering algorithm being evaluated. CARP is released under the GNU GPL license.

## 1.1 Project homepage

The project homepage is located on the Worldwide Web at:

- <http://www.public.iastate.edu/~maitra/Software/CARP.html>

The webpage contains instructions, an updated version of this manual and the source code in a zipped format.

## 1.2 Dependencies

CARP has the following dependencies required to be installed before installing CARP:

- gcc compiler
- glibc library
- other programming language and packages if needed for an external clustering algorithm or clustering metrics

# 2 Installation

CARP can be installed in two easy steps:

1. Extract files from CARP\_v3.3.tar.gz using  
`tar -xzf CARP_v3.3.tar.gz`

2. Compile the files running the command  
`make CARP`

This creates the executable files: `CARP` and `unittest`. Files `AdjRand`, `Rand`, `VarInf`, `ProAgree`, `Diag`, `BHI`, `Kmeans`, `Mclust`, `PAM`, and `hierclust` will be also created; however, they are optional files and the command to create either may be removed from the included `makefile` if needed.

To check the integrity of the package, run the command

- `make check`

To remove the installed files, use the command

- `make clean`

## 3 Usage

### 3.1 Parameters

Running `CARP` involves a simple command of the following form:

```
./CARP set-of-parameters
```

The following parameters can be specified in the above command line:

- b: average overlap (no default value)
- m: maximum overlap (no default value)
- g: generalized overlap (no default value)
- p: number of dimensions (2 by default)
- G: number of mixing components at simulation stage (2 by default)
- s: spherical covariance matrix structure (non-spherical by default if option is unspecified)
- H: homogeneous covariance matrices (non-homogeneous by default if option is unspecified)
- e: maximum eccentricity (0.90 by default)
- z: smallest mixing proportion (equal mixing proportions  $1/K$  by default)
- k: minimum number of mixing components (no default value)
- K: maximum number of mixing components (no default value)
- L: lower bound for Uniform(*lower-bound*, *upper-bound*) distribution from which mean vectors are generated (0 by default)
- U: upper bound for Uniform(*lower-bound*, *upper-bound*) distribution from which mean vectors are generated (1 by default)
- r: maximum number of resimulations (1000 by default)
- a: accuracy of estimation (1e-06 by default)
- l: maximum number of integration terms used by `qfc` function (1e06 by default)
- P: file containing mixing proportions (`Pi.dat` by default)
- M: file containing mean vectors (`Mu.dat` by default)
- S: file containing covariance matrices in triangular form (`LTSigma.dat` by default)
- D: working directory (`DATA` by default)

- N: file containing cluster sizes (**Nk.dat** by default)
- I: file containing true classifications (**idTrue.dat** by default)
- i: file containing estimated classifications (**idEst.dat** by default)
- X: file containing simulated datasets (**x.dat** by default)
- W: file containing maps of pairwise overlaps (**overMap.dat** by default)
- O: file containing characteristics of simulated mixtures (**overBarMax.dat** by default)
- R: file containing index values (adjusted Rand index and **AR.dat** by default)
- n: number of observations generated from every mixture (0 by default)
- #: number of simulated mixtures (1 by default)
- V: file containing coefficients for the inverse Box-Cox transformation
- w: number of noise coordinates (0 by default)
- o: number of outliers (0 by default)
- A: algorithm (command) to be used for clustering (if this option is not specified, CARP only simulates mixtures and/or datasets)
- c: use the command line interface with option -A (FALSE by default)
- E: evaluation (command) to be used for evaluating partitioning algorithm (if this option is not specified, AdjRand is run)
- C: use the command line interface with option -E (FALSE by default)
- h: help.

## 3.2 Details

### 3.2.1 Organization of Files

The files are provided in the following sub-directories:

- **src** directory contains all the C files and the header files.
- **PROC** directory contains all the files contained in the examples.
- **TEST** directory contains test programs to check the integrity of the package, as well as files needed in the example illustration of the Iris dataset.
- **DATA** is the working directory. All files created in the simulation and evaluation steps are stored here.

### 3.2.2 Arguments and Options for CARP

Upon launching CARP, three stages of the program are invoked in sequence. The first stage is responsible for the simulation of Gaussian mixtures with pre-specified level of complexity expressed in terms of pairwise overlap and for the generation of datasets from these mixtures. The following options are related to the first stage: -b, -m, -g, -p, -G, -s, -H, -e, -z, -L, -U, -r, -a, -l, -P, -M, -S, -D, -N, -I, -X, -W, -O, -n, -#, -V, -w, -o. The option -h provides help. The output will be directed to a working folder specified by the option -D. The obtained parameters – mixing proportions, mean vectors and covariance matrices – will be saved to the files specified by the options -P, -M, and -S correspondingly while samples drawn from the mixtures will be saved into the file specified by the option -X. The file provided with the

option `-I` contains true classifications for all observations while option `-N` provides the name for the file with cluster sizes. In addition, the map of misclassification probabilities will be saved into the file specified by the option `-W` while average and maximum overlaps as well as the row and column numbers of the components that produced maximum overlap are stored in the file given by the option `-C`. The element with the index  $(i, j)$  in the misclassification map represents the probability that  $X$  simulated from the  $i$ th component is classified to the  $j$ th component.

At the second stage, a user-specified clustering method has to be run for the simulated datasets. For illustration, several examples involving algorithms such as hierarchical clustering with Ward's linkage (Ward, 1963) (`hierclust`), expectation-maximization (Dempster et al., 1977; Fraley and Raftery, 2006) (`Mclust`),  $k$ -means (Forgy, 1965; MacQueen, 1967) (`Kmeans`) as implemented by Hartigan and Wong (1979), and partitioning around medoids (Kaufman and Rousseeuw, 1990) (`PAM`) are provided. Other methods can be run with the option `-A`. The user's clustering program has to be written to accept the following options: `-p`, `-n`, `-#`, `-D`, `-i`, `-X`, `-K`, and also `-k` if the best clustering solution has to be searched between the minimum and maximum numbers of clusters specified by options `-k` and `-K` respectively. The program will obtain partitionings and write them into the file specified by the option `-i`. Two options are provided, one for the expert user and the other for those that are not so comfortable with writing code. For the expert user, it is assumed that (s)he is comfortable writing a small piece of code responsible for accepting the above parameters coming from the first stage. The approach is illustrated in Examples 4.6.1 and 4.7.1. For the less-experienced user not comfortable with writing a few lines of C code, the command-line interface provides a ready alternative; (s)he can invoke the option `-c` included into the call of CARP. In this case, CARP will not attempt to pass the parameters between stages. The user is then responsible for specifying these parameters within a clustering program or command line. For more details, see Examples 4.6.2, 4.6.3, 4.7.2, and 4.7.3.

At the third stage, classifications derived by the clustering algorithm under investigation are compared with the true to assess performance. The adjusted Rand index (`AdjRand`) is incorporated as a default measure of similarity between the two classifications. Two other options provided in CARP are Meilă (2006)'s variation of information (`VarInf`) and the proportion of correctly classified observations (`ProAgree`). An user, however, can specify other programs for comparing the obtained and true partitionings. Examples of other methods that are included with this manual are through calls to R packages which calculate the Rand index (Rand, 1971) (`Rand`), the biological homogeneity index (Datta and Datta, 2006) (`BHI`), and the proportion of correctly classified observations (`Diag`). If an option `-E` is specified, the program should comply with the following options: `-n`, `-#`, `-D`, `-I`, `-i`, `-R`. Calculated values of the provided similarity measure have to be written to the file specified by the option `-R`. For details, see Examples 4.8.1 and 4.9.1. Alternatively, less-experienced users can specify an option `-C` with similar meaning as for the option `-c` from the second stage: a user can provide a command line invoking a partitioning-analyzing program. For more details, see Examples 4.8.2, 4.8.3, 4.9.2, and 4.9.3.

If options `-b` and `-m` are both specified (see Example 4.1), CARP produces a mixture satisfying both the characteristics of average and maximum overlap. If one option, `-b`, `-m`, or `-g`, is specified (see Example 4.2), a mixture satisfying the pre-specified value is generated. While the notion of maximum and average overlap is self-explanatory (or see Maitra and

Melnykov (2010b)), the generalized overlap is defined as

$$\ddot{\omega} = \frac{\lambda_{(K)} - 1}{K - 1},$$

where  $K$  represents the desired number of groups or mixture components and  $\lambda_{(K)}$  is the largest eigenvalue of the symmetric  $K \times K$ -dimensional matrix  $\Omega$  defined as the sum of the matrix of misclassification probabilities  $\Omega_{i|j}$  and its transpose:

$$\Omega = \Omega_{i|j} + \Omega'_{i|j},$$

where

$$\Omega_{i|j} = \begin{pmatrix} 1 & \omega_{2|1} & \omega_{3|1} & \dots & \omega_{K|1} \\ \omega_{1|2} & 1 & \omega_{3|2} & \dots & \omega_{K|2} \\ \omega_{1|3} & \omega_{2|3} & 1 & \dots & \omega_{K|3} \\ \dots & \dots & \dots & \dots & \dots \\ \omega_{1|K} & \omega_{2|K} & \omega_{3|K} & \dots & 1 \end{pmatrix}$$

with  $\omega_{i|j}$  representing the probability that an observation originated from the  $j$ th component but was misclassified to the  $i$ th component.

The generalized overlap was proposed by Maitra (2010) (refer to that paper for more details) as a way to summarize the overlap in detected activation in several fMRI studies. The idea is easily adapted for our case: we advocate its use as a more useful single measure than either the average or the maximum overlap. This is particularly important in the context of simulating homogeneous spherical distributions where there are not enough free parameters to control both the average and the maximum overlap. Appendix B provides some details on geometry of mixture distributions simulated according to pre-specified levels of generalized, maximum, and average overlap.

### 3.3 Other capabilities

#### 3.3.1 Some useful features

If none of the options `-b`, `-m`, and `-g` are provided, CARP reads parameters from the files specified by the options `-P`, `-M`, `-S` and computes misclassification probabilities for the components of the given mixture model. Note that the options `-p` and `-G` have to be correctly specified. The options `-n` and `-#` specify the sample size of a sample drawn from every generated mixture and the number of such mixtures correspondingly. If it is desired to use datasets stored in a file specified by the option `-X`, the sample size should be given as a negative number, for example: `-n-100` instead of the traditional `-n100`. In that case, new datasets will not be simulated. Note that this capability is available only in the mode when mixture parameters are not simulated but rather read from files. This capability can be useful in the case when it is desired to run several clustering algorithms on the same set of simulated data. Example 4.3 provides an illustration of these features.

Another application of CARP's capabilities is for assessing the level of clustering complexity of derived and existing groups in datasets based on the set of estimated mixture model parameters. Example 4.4 provides an illustration on the use of this feature for learning the properties of the celebrated *Iris* dataset (Anderson, 1935; Fisher, 1936).

### 3.3.2 Using CARP standalone to generate mixture models and datasets

CARP also has a capability to simulate finite mixture models and datasets from them without launching the second and third stages. Thus, the simulation stage would be the only one to run. In order to do so, the option `-A` (and `-E`) has to be omitted. CARP will run as illustrated in Example 4.5.

Note that this functionality is also provided by the R package `MixSim` whose details and parameters are provided in the appendix.

### 3.3.3 Using external clustering procedures

It might be desirable for an user to run a clustering algorithm written in a language other than C. CARP easily handles such situations with the options `-A` and `-c`. The latter one allows switching to a command line interface as described in Section 3.2.2. Note that it is the user's responsibility to ensure that the other language is installed and all necessary programs compiled prior to running CARP. Examples 4.6 and 4.7 illustrate this process running various procedures in different ways.

### 3.3.4 Using external metrics for assessing performance of clustering procedures

By default, CARP uses the adjusted Rand index for assessing clustering performance. Similar to the previous section, it might be desired to use some external function written in another language. For instance, R packages `e1071` and `clValid` that contain several metrics that can be employed to evaluate the performance of clustering algorithms. Examples 4.8 and 4.9 provide illustrations for dealing with these two packages using options `-E` and `-C`. Some other metrics, such as ones considered in Meilă (2006) and included in the package `MixSim` (Appendix A), can be used similarly.

### 3.3.5 Simulating non-Gaussian mixtures

By default, CARP generates datasets from Gaussian mixture models. However, it is not constrained to be so, for CARP also allows for simulating non-Gaussian datasets with desired approximate clustering complexity. The underlying procedure uses the multivariate Box-Cox transformation given by

$$y^* = \frac{y^\lambda - 1}{\lambda},$$

where  $y$  and  $y^*$  represent the original and transformed observation, respectively. Obtained set of  $y^*$ s is approximately normally distributed for some specific value of  $\lambda$ . CARP uses an inverse Box-Cox transformation producing non-Gaussian realizations from normally distributed data. The transformation is given by

$$y = (\lambda y^* + 1)^{\frac{1}{\lambda}}.$$

Since  $\lambda = 1$  leads to the reduced formula  $y = y^* + 1$  which does not eliminate normality in data providing just a shift to  $y^*$ , we use the formula

$$y = (\lambda y^* + 1)^{\frac{1}{\lambda}} - 1.$$

Note also that  $\lambda = 1$  yields the identity transformation. Desired transformations should be specified in the file provided with the option `-V`. Every coordinate is transformed independent from the others. If a transformation coefficient for some of the coordinates is not specified, it is assumed that no transformation is needed for that coordinate, *i.e.*  $\lambda = 1$  for that coordinate. For example, assume that a four-dimensional dataset is needed to be generated with the first and third coordinates being transformed with  $\lambda = 0.5$ . Then, a file should be created with the following content: 0.5 1.0 0.5. If the file specified by the option `-V` does not exist, random transformations with  $\lambda \in (0, 1)$  will be employed and file with coefficients will be created. Example 4.10 illustrates the use of the option `-V`.

### 3.3.6 Simulating noisy coordinates

Clustering can be challenging in the presence of coordinates containing no clustering information. CARP allows simulating such noisy coordinates that are pseudo-random deviates uniformly distributed on a hypercube specified by lower and upper bounds (options `-L` and `-U`). The number of desired noisy coordinates is specified with option `-w`. See Example 4.11 for an illustration.

### 3.3.7 Simulating datasets with outliers

Performance of clustering procedures can be severely affected by the presence of outliers. CARP defines outliers as observations lying outside all 99.9% hyperellipsoidal contours associated with covariance matrices. The required number of outliers is specified with the option `-o`. Example 4.12 illustrates this capability.

## 4 Illustrations

### 4.1 Example: Evaluating clustering algorithms with spherical dispersion structures and equal representation

This example illustrates the simulation of three 2-dimensional 4-component mixtures with spherical covariance matrices and equal mixing proportions according to pre-specified levels of maximum and average overlap, drawing samples of 100 observations from simulated mixtures, running the clustering program `hierclust`, and computing the adjusted Rand index. Consider the following command:

```
./CARP -m0.1 -b0.05 -p2 -G4 -s -n100 -#3 -DDATA -Xdata.dat -IIDtrue.dat \
-iIIDest.dat -RAAdjRand.dat -Ahierclust -r1000
```

CARP runs the following three steps:

1. *Mixture simulation*

Three 2-dimensional 4-component mixtures with spherical covariance matrices, equal mixing proportions, maximum overlap 0.1, and average overlap 0.05 are simulated. 1000 resimulations are allowed for reaching the desired level of overlap. 100 observations will be generated from every mixture. The working directory is `DATA`. Simulated



samples will be stored into the file `data.dat` while the classifications will be stored into the file `IDtrue.dat`. Files with mixture parameters keep their default names specified in the options description (Section 3.1). The corresponding output is provided below.

### Output:

#### MIXTURE MODEL #1:

The desired overlap has been met...

Map of misclassification probabilities:

[0][0] : 1.000000 [0][1] : 0.015200 [0][2] : 0.003266 [0][3] : 0.033826

[1][0] : 0.012737 [1][1] : 1.000000 [1][2] : 0.000054 [1][3] : 0.035891

[2][0] : 0.003856 [2][1] : 0.000074 [2][2] : 1.000000 [2][3] : 0.036315

[3][0] : 0.049968 [3][1] : 0.064109 [3][2] : 0.044703 [3][3] : 1.000000

Average Overlap: 0.050000

Maximum Overlap: 0.100000 (components: 1 and 3)

Generalized Overlap: 0.055460

Mixture parameters:

Pi:

0.250000 0.250000 0.250000 0.250000

Mu:

[0] : 0.471083 0.675651

[1] : 0.335962 0.337816

[2] : 0.980450 0.580291

[3] : 0.652717 0.362861

Sigma:

[0] :

0.007961 0.000000

0.000000 0.007961

[1] :

0.005868 0.000000

0.000000 0.005868

[2] :

0.010726 0.000000

0.000000 0.010726

[3] :

0.015040 0.000000

0.000000 0.015040

Dataset with cluster sizes  $N_k = 28 \ 21 \ 29 \ 22$  has been generated...

#### MIXTURE MODEL #2:

The desired overlap has been met...

Map of misclassification probabilities:

[0][0] : 1.000000 [0][1] : 0.030979 [0][2] : 0.029165 [0][3] : 0.001868  
[1][0] : 0.013597 [1][1] : 1.000000 [1][2] : 0.010535 [1][3] : 0.006902  
[2][0] : 0.048703 [2][1] : 0.036413 [2][2] : 1.000000 [2][3] : 0.056377  
[3][0] : 0.002374 [3][1] : 0.019464 [3][2] : 0.043623 [3][3] : 1.000000

Average Overlap: 0.050000

Maximum Overlap: 0.100000 (components: 2 and 3)

Generalized Overlap: 0.051446

Mixture parameters:

Pi:

0.250000 0.250000 0.250000 0.250000

Mu:

[0] : 0.203039 0.189259

[1] : 0.485454 0.285278

[2] : 0.338648 0.623985

[3] : 0.795138 0.499927

Sigma:

[0] :

0.010917 0.000000

0.000000 0.010917

[1] :

0.002621 0.000000

0.000000 0.002621

[2] :

0.025417 0.000000

0.000000 0.025417

[3] :

0.016861 0.000000

0.000000 0.016861

Dataset with cluster sizes Nk = 26 22 21 31 has been generated...

MIXTURE MODEL #3:

The desired overlap has been met...

Map of misclassification probabilities:

[0][0] : 1.000000 [0][1] : 0.000000 [0][2] : 0.021907 [0][3] : 0.020430  
[1][0] : 0.000000 [1][1] : 1.000000 [1][2] : 0.013279 [1][3] : 0.000000  
[2][0] : 0.078093 [2][1] : 0.051412 [2][2] : 1.000000 [2][3] : 0.041697  
[3][0] : 0.049043 [3][1] : 0.000000 [3][2] : 0.024139 [3][3] : 1.000000

Average Overlap: 0.050000

Maximum Overlap: 0.100000 (components: 0 and 2)

Generalized Overlap: 0.051418







### 3. *Computing adjusted Rand index*

```
./AdjRand -n100
```

The above command computes the adjusted Rand index and writes it into the default file `AR.dat`.

#### **Output:**

```
ADJUSTED RAND VALUES:  
Ktrue = 4 Kest = 4 : 1.000000
```

## 4.3 Example: Evaluating a clustering algorithm from data simulated earlier and stored in files

This example is on reading a 2-dimensional 4-component mixture and a dataset of size 100 from files produced in Example 4.2, running the clustering program `hierclust`, and computing the adjusted Rand index.

Consider the following command:

```
./CARP -p2 -G4 -n-100 -Ahierclust
```

This command reads parameters and one dataset obtained in Example 4.2 from the default files (see Section 3.1) and runs `hierclust`. After that, the adjusted Rand index is computed.

### 1. *Mixture simulation*

Here, `CARP` reads one 2-dimensional 4-component mixture from the default files for mixing proportions, means, and dispersions. 100 observations are read from the default file `x.dat`.

#### **Output:**

```
The desired overlap has been met...  
Map of misclassification probabilities:  
[0][0] : 1.000000 [0][1] : 0.000000 [0][2] : 0.000000 [0][3] : 0.000921  
[1][0] : 0.000000 [1][1] : 1.000000 [1][2] : 0.000000 [1][3] : 0.000000  
[2][0] : 0.000000 [2][1] : 0.000000 [2][2] : 1.000000 [2][3] : 0.000000  
[3][0] : 0.000079 [3][1] : 0.000000 [3][2] : 0.000000 [3][3] : 1.000000  
Average Overlap: 0.000167  
Maximum Overlap: 0.001000 (components: 0 and 3)  
Generalized Overlap: 0.000334
```

### 2. *Running clustering algorithm*

```
./hierclust -p2 -K4 -n-100
```

This command runs the program `hierclust` which performs hierarchical clustering.

Output:

OBTAINED CLASSIFICATIONS:

[illegible]

### 3. Computing adjusted Rand index

```
./AdjRand -n-100
```

The above command computes the adjusted Rand index and writes it into the default file `AR.dat`.

Output:

ADJUSTED RAND VALUES:

Ktrue = 4 Kest = 4 : 1.000000

#### 4.4 Example: Calculating overlap between identified classes in a dataset

Here we provide an example that illustrates the use of the program in calculating the overlap between identified groups in a dataset, such as the celebrated *Iris* dataset (included). Consider the following command:

```
./CARP -p4 -G3 -D"TEST" -P"Pi.iris" -M"Mu.iris" -S"LTSigma.iris"
```

Here we provide an illustration based on the famous *Iris* dataset. Files `Pi.iris`, `Mu.iris`, and `LTSigma.iris` at the directory `TEST` contain parameters of the Gaussian mixture model with 3 components computed based on the true classification. `CARP` runs only the first stage evaluating pairwise overlaps for provided parameters. The second and third stages will not be run as the option `-A` is not specified. No data are simulated as the option `-n` is not provided.

### 1. *Reading mixture*

The command reads parameters for *Iris*. From the output, it is clearly seen that the second and third components of the mixture produce substantial overlap while the other two pairs do not yield any noticeable overlap.

Output:

Map of misclassification probabilities:

```

[0][0] : 1.000000 [0][1] : 0.000000 [0][2] : 0.000000
[1][0] : 0.000000 [1][1] : 1.000000 [1][2] : 0.023023
[2][0] : 0.000000 [2][1] : 0.026294 [2][2] : 1.000000
Average Overlap: 0.016439
Maximum Overlap: 0.049318 (components: 1 and 2)
Generalized Overlap: 0.024658

```

## 4.5 Example: Use of CARP as a standalone package

Here, we illustrate the use of CARP standalone, specifically in the case of simulating two 2-dimensional 2-component mixtures with general covariance matrices and mixing proportions no less than 0.3 according to an average overlap of 0.2 and generating datasets of size 100. Consider the following command:

```
./CARP -b0.2 -p2 -G2 -\#2 -n100 -z0.3
```

Two 2-dimensional 2-component mixtures will be generated according to the average overlap value of 0.2 and samples of size 100 will be drawn from them. CARP runs only the simulation stage as the option `-A` is not specified. The files will have their default names specified in Section 3.1. The smallest mixing proportion allowed is 0.3. As there are only two components, specifying the value of average overlap is equivalent to providing that of maximum overlap.

### 1. *Mixture simulation*

Output from the program is given below.

#### **Output:**

MIXTURE MODEL #1:

The desired overlap has been met...

Map of misclassification probabilities:

```

[0][0] : 1.000000 [0][1] : 0.065789
[1][0] : 0.134211 [1][1] : 1.000000
Average Overlap: 0.200000
Maximum Overlap: 0.200000 (components: 0 and 1)
Generalized Overlap: 0.200000

```

Mixture parameters:

Pi:

0.476254 0.523746

Mu:

[0] : 0.482694 0.868947

[1] : 0.228367 0.105629



```

Sigma:
[0] :
0.068956 - 0.061422
-0.061422 0.138259
[1] :
0.057771 0.007065
0.007065 0.216991
Dataset with cluster sizes Nk = 53 47 has been generated...

```

MIXTURE MODEL #2:

```

The desired overlap has been met...
Map of misclassification probabilities:
[0][0] : 1.000000 [0][1] : 0.167245
[1][0] : 0.032755 [1][1] : 1.000000
Average Overlap: 0.200000
Maximum Overlap: 0.200000 (components: 0 and 1)
Generalized Overlap: 0.200000

```

Mixture parameters:

```

Pi:
0.315557 0.684443
Mu:
[0] : 0.290308 0.104932
[1] : 0.360821 0.169447
Sigma:
[0] :
0.001618 0.000344
0.000344 0.001961
[1] :
0.001450 - 0.001001
-0.001001 0.002347
Dataset with cluster sizes Nk = 28 72 has been generated...

```

## 4.6 Example: Using external clustering programs - the R function PAM

**Note:** This example requires the R package `cluster` to be installed by the user.

### 4.6.1 Expert Mode for the Advanced User

Here, we provide an example for the advanced user on how to use external programs written in another language. We consider a R function `pam` from the R package `cluster`. This

procedure is a realization of partitioning around medoids. A small program `PAM.c` available in the main package directory is written in C. It analyzes necessary parameters and passes them to a R function by running the following piece of code.

```
/* Run PAM clustering */
sprintf(runR, "R CMD BATCH --no-save --no-restore '--args %d %d %d %d %s %s'
PROC/runPAM.R", p, K, g, nf, Xname, IDname);
code = system(runR);
```

Here, the parameters  $p$ ,  $K$ ,  $g$ ,  $nf$ ,  $Xname$ ,  $IDname$  are passed to the R program `runPAM.R`. `runPAM.R` is located in the directory `PROC`, its code is provided below.

```
PAM.call <- function(p, K, n, nf, Xname, IDname){

for (i in 1:nf){

x <- matrix(scan(Xname, nlines = n, skip = (i - 1) * n), ncol = p, byrow = T)

A <- pam(x, K)

if (i == 1) write(x = A$clustering - 1, file = IDname, ncol = n)
else write(x = A$clustering - 1, file = IDname, ncol = n, append = TRUE)

}
}

args = (commandArgs(TRUE))

p <- as.numeric(args[1])
K <- as.numeric(args[2])
n <- as.numeric(args[3])
nf <- as.numeric(args[4])
Xname <- args[5]
IDname <- args[6]

if (!require(cluster)) stop("Package cluster is not installed...")

if (inherits(try(PAM.call(p, K, n, nf, Xname, IDname)),
what = "try-error"))
stop("Problem with PAM.call; check the parameters...")
```

Consider the following command:

```
./CARP -m0.2 -p3 -G4 -n100 -APAM
```

A three-dimensional mixture with 4 components will be generated. The desired maximum overlap is 0.2. 100 observations have to be simulated and the clustering program PAM will be run. The results should be analyzed by the adjusted Rand index. The default file and directory names are used (see Section 3.1).

# 1. *Mixture simulation*

As in the previous examples, CARP simulates a desired mixture model with specified characteristics.

## Output:

The desired overlap has been met...

Map of misclassification probabilities:

```
[0][0] : 1.000000 [0][1] : 0.039084 [0][2] : 0.003227 [0][3] : 0.017701
[1][0] : 0.055826 [1][1] : 1.000000 [1][2] : 0.048964 [1][3] : 0.094561
[2][0] : 0.001574 [2][1] : 0.021713 [2][2] : 1.000000 [2][3] : 0.046037
[3][0] : 0.018637 [3][1] : 0.105439 [3][2] : 0.077478 [3][3] : 1.000000
```

Average Overlap: 0.088373

Maximum Overlap: 0.200000 (components: 1 and 3)

Generalized Overlap: 0.078662

Mixture parameters:

Pi:

0.250000 0.250000 0.250000 0.250000

Mu:

[0] : 0.107773 0.116045 0.473192

[1] : 0.409403 0.237800 0.978207

[2] : 0.153951 0.827237 0.918648

[3] : 0.430444 0.703625 0.550648

Sigma:

[0] :

0.042778 - 0.016666 - 0.030209

-0.016666 0.054582 - 0.003463

-0.030209 - 0.003463 0.080161

[1] :

0.095056 - 0.037666 - 0.053635

-0.037666 0.062020 0.017224

-0.053635 0.017224 0.109266

[2] :

0.026073 0.005839 - 0.010612

0.005839 0.016476 - 0.010769

-0.010612 - 0.010769 0.021628

[3] :

0.027854 0.009114 - 0.018176

```
0.009114 0.040695 - 0.036502
-0.018176 - 0.036502 0.097819
Dataset with cluster sizes Nk = 23 24 27 26 has been generated...
```

## 2. *Running PAM*

At this stage, the program PAM will be run as described above.

```
./PAM -p3 -n100 -K4
```

### **Output:**

OBTAINED CLASSIFICATIONS:

```
0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 2 2 1 2 2 2 2 2 1 2 2 1 2 3
2 2 0 2 2 2 2 3 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 3 3 3 3 3 3 1 3 0 3 3 1 3 3 3 3 3 1 2 3 2 3 3 3 2
```

The obtained estimated classification is written into the default file DATA/idEst.dat.

## 3. *Calculating adjusted Rand index*

Finally, the adjusted Rand index is calculated and the result is saved in AR.dat in the DATA directory.

```
./AdjRand -n100
```

### **Output:**

ADJUSTED RAND VALUES:

```
Ktrue = 4 Kest = 4 : 0.588823
```

## 4.6.2 Command-Line interface for the Less-Experienced User

In many situations, the user might prefer to use a command line interface without dealing with writing or modifying some code and passing parameters as in Example 4.6.1. The following example illustrates such a situation. The code of PAM.R is provided below.

```
PAM.call <- function(p, K, n, nf, Xname, IDname){
  for (i in 1:nf){
    x <- matrix(scan(Xname, nlines = n, skip = (i - 1) * n), ncol = p, byrow = T)
    A <- pam(x, K)
    if (i == 1) write(x = A$clustering - 1, file = IDname, ncol = n)
    else write(x = A$clustering - 1, file = IDname, ncol = n, append = TRUE)
```

```

}

}

p <- 3
K <- 4
n <- 100
nf <- 1
Xname <- "DATA/x.dat"
IDname <- "DATA/idEst.dat"

if (!require(cluster)) stop("Package cluster is not installed...")

if (inherits(try(PAM.call(p, K, n, nf, Xname, IDname)), what = "try-error"))
stop("Problem with PAM.call; check the parameters...")

```

The following command has to be run.

```
./CARP -m0.2 -p3 -G4 -n100 -c -A"R CMD BATCH PROC/PAM.R"
```

Here, a three-dimensional mixture with 4 components will be generated. The desired maximum overlap is 0.2. 100 observations have to be simulated and the clustering program PROC/PAM.R will be run. The option `-c` implies that the command line interface has to be employed. The results should be analyzed by the adjusted Rand index.

### 1. *Mixture simulation*

#### **Output:**

```

The desired overlap has been met...
Map of misclassification probabilities:
[0][0] : 1.000000 [0][1] : 0.069034 [0][2] : 0.026670 [0][3] : 0.000285
[1][0] : 0.130966 [1][1] : 1.000000 [1][2] : 0.013417 [1][3] : 0.000555
[2][0] : 0.044243 [2][1] : 0.014339 [2][2] : 1.000000 [2][3] : 0.029365
[3][0] : 0.000344 [3][1] : 0.000372 [3][2] : 0.043560 [3][3] : 1.000000
Average Overlap:  0.062192
Maximum Overlap:  0.200000 (components:  0 and 1)
Generalized Overlap:  0.076326

Mixture parameters:
Pi:
0.250000 0.250000 0.250000 0.250000
Mu:
[0] : 0.343234 0.744296 0.335899
[1] : 0.265788 0.808462 0.638652

```

```

[2] : 0.116432 0.284932 0.353443
[3] : 0.315915 0.040121 0.546591
Sigma:
[0] :
0.030450 0.007690 - 0.007662
0.007690 0.010043 - 0.000884
-0.007662 - 0.000884 0.012413
[1] :
0.064435 0.011950 - 0.013849
0.011950 0.033798 - 0.027993
-0.013849 - 0.027993 0.069561
[2] :
0.008674 0.007417 0.002020
0.007417 0.028284 0.006177
0.002020 0.006177 0.008199
[3] :
0.014454 - 0.007461 0.003208
-0.007461 0.012601 - 0.002027
0.003208 - 0.002027 0.005187
Dataset with cluster sizes Nk = 21 22 25 32 has been generated...

```

## 2. *Running* R CMD BATCH PROC/PAM.R

Runs the program PAM.R from the directory PROC. The obtained estimated classification is written into the default file DATA/idEst.dat.

## 3. *Running* AdjRand

```
./AdjRand -n100
```

The adjusted Rand index is calculated and saved in DATA/AR.dat.

### Output:

```

ADJUSTED RAND VALUES:
Ktrue = 4 Kest = 4 : 0.691559

```

### 4.6.3 Command-Line Interface: An alternative approach

An alternative approach similar to the one provided above in Section 4.6.2 is to employ the command

```

./CARP -m0.2 -p3 -G4 -n100 -c -A"R CMD BATCH --no-save \
--no-restore '--args 3 4 100 1 DATA/x.dat DATA/idEst.dat' \

```

PROC/runPAM.R"

Here, parameter values are not provided in the file being specified in the command line and passed to PROC/runPAM.R. The output file is as in Example 4.6.2.

## 4.7 Example: Using external clustering procedures - the R function Mclust

**Note:** This example requires the R package `mclust` to be installed.

### 4.7.1 Expert Mode for the Advanced User

The same approach can be used for running other clustering procedures. This example, however, is different from Example 4.6. It illustrates how another popular R procedure `Mclust` from the package `mclust` can be employed in the case when the number of clusters is assumed to be unknown and has to be estimated at stage 2. The program `Mclust.c` is almost identical to `PAM.c` from Example 4.6.1 with the difference in the small piece of code given below.

```
/* Run Mclust clustering */
sprintf(runR, "R CMD BATCH --no-save --no-restore '--args %d %d %d %d %d %s %s'
runMclust.R", p, k, K, g, nf, Xname, IDname);
code = system(runR);
```

Here, we launch the external program `runMclust.R` which will be run with all necessary parameters passed to it. We should note that instead of passing the number of clusters, we pass the minimum and maximum numbers of clusters from the range on which the clustering procedure has to be run. The code of `runMclust.R` is provided below.

```
Mclust.call <- function(p, k, K, n, nf, Xname, IDname){

for (i in 1:nf){

x <- matrix(scan(Xname, nlines = n, skip = (i - 1) * n), ncol = p, byrow = T)

A <- Mclust(x, G = k:K, model = "VVV")

if (i == 1) write(x = A$class - 1, file = IDname, ncol = n)
else write(x = A$class - 1, file = IDname, ncol = n, append = TRUE)

}

}

args = (commandArgs(TRUE))
```

```

p <- as.numeric(args[1])
k <- as.numeric(args[2])
K <- as.numeric(args[3])
n <- as.numeric(args[4])
nf <- as.numeric(args[5])
Xname <- args[6]
IDname <- args[7]

if (!require(mclust)) stop("Package mclust is not installed...")

if (inherits(try(Mclust.call(p, k, K, n, nf, Xname, IDname)),
what = "try-error"))
stop("Problem with Mclust.call; check the parameters...")

```

Consider the following command.

```
./CARP -b0.1 -p3 -G4 -n100 -k1 -K7 -AMclust
```

The command is similar to that from the Example 4.6.1 with the difference that **Mclust** is employed in place of **PAM** and options **-k** and **-K** are specified. Also, the average overlap 0.1 is employed.

#### 1. *Mixture simulation*

##### **Output:**

```

The desired overlap has been met...
Map of misclassification probabilities:
[0][0] : 1.000000 [0][1] : 0.044793 [0][2] : 0.029921 [0][3] : 0.063393
[1][0] : 0.058305 [1][1] : 1.000000 [1][2] : 0.026801 [1][3] : 0.047219
[2][0] : 0.027978 [2][1] : 0.057768 [2][2] : 1.000000 [2][3] : 0.014438
[3][0] : 0.134094 [3][1] : 0.073647 [3][2] : 0.021644 [3][3] : 1.000000
Average Overlap:  0.100000
Maximum Overlap:  0.197487 (components:  0 and 3)
Generalized Overlap:  0.102172

Mixture parameters:
Pi:
0.250000 0.250000 0.250000 0.250000
Mu:
[0] : 0.111050 0.612100 0.571633
[1] : 0.517080 0.615308 0.360086
[2] : 0.265869 0.168437 0.651138
[3] : 0.125562 0.815075 0.808542

```



```

Sigma:
[0] :
0.018757 - 0.008896 - 0.003740
-0.008896 0.012202 - 0.001916
-0.003740 - 0.001916 0.026191
[1] :
0.038855 - 0.019627 - 0.002308
-0.019627 0.031330 0.002852
-0.002308 0.002852 0.010926
[2] :
0.045396 0.017788 - 0.021713
0.017788 0.024984 - 0.008391
-0.021713 - 0.008391 0.040534
[3] :
0.034761 - 0.016823 - 0.002626
-0.016823 0.040276 - 0.018062
-0.002626 - 0.018062 0.085983
Dataset with cluster sizes Nk = 18 32 28 22 has been generated...

```

## 2. *Running Mclust*

```
./Mclust -k1 -K7 -p3 -n100
```

Here, `Mclust` detects the best model according to BIC. Models being tested have from 1 to 7 mixture components.

### **Output:**

OBTAINED CLASSIFICATIONS:

```

0 0 0 0 0 1 1 0 1 0 0 1 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 1 0 1 1 0 0 1 1 1 1 1 1 1 0 1 1 1 1 1 1
0 1 1 1 1 0 1 0 0 1 1 1 1 0 1 1 1 0 0 0 1 1 0 0 1 1

```

As we can see, according to `Mclust`, the best model has two components. The estimated classification is written into the default file `DATA/idEst.dat`.

## 3. *Running AdjRand*

```
./AdjRand -n100
```

This command calculates the adjusted Rand index and saves its value to `AR.dat` in the directory `DATA`.

### **Output:**

ADJUSTED RAND VALUES:

Ktrue = 4 Kest = 2 : 0.194548

#### 4.7.2 Command-Line interface for the Less-Experienced User

This is another illustration of the command-line interface. Upon generating one three-dimensional mixture with 4 components with maximum overlap 0.1 and simulating a dataset of size 100 from it, the user wants to run his or her own function `Mclust.R` from the directory `PROC` varying the number of clusters. Then, the adjusted Rand value will be calculated for the detected number of clusters. The code of `Mclust.R` is provided below.

```
Mclust.call <- function(p, k, K, n, nf, Xname, IDname){  
  
  for (i in 1:nf){  
  
    x <- matrix(scan(Xname, nlines = n, skip = (i - 1) * n), ncol = p, byrow = T)  
  
    A <- Mclust(x, G = k:K, model = "VVV")  
  
    if (i == 1) write(x = A$class - 1, file = IDname, ncol = n)  
    else write(x = A$class - 1, file = IDname, ncol = n, append = TRUE)  
  
  }  
}  
  
p <- 3  
k <- 1  
K <- 7  
n <- 100  
nf <- 1  
Xname <- "DATA/x.dat"  
IDname <- "DATA/idEst.dat"  
  
if (!require(mclust)) stop("Package mclust is not installed...")  
  
if (inherits(try(Mclust.call(p, k, K, n, nf, Xname, IDname)),  
  what = "try-error"))  
  stop("Problem with Mclust.call; check the parameters...")
```

The following command has to be run.

```
./CARP -m0.1 -p3 -G4 -n100 -k1 -K7 -c -A"R CMD BATCH PROC/Mclust.R"
```

A three-dimensional mixture with 4 components will be generated. The desired maximum overlap is 0.1. 100 observations have to be simulated and the clustering program `PROC/Mclust.R` will be run. The option `-c` implies that the command line interface has

to be employed. The number of clusters to try is between 1 and 7. The results should be analyzed by the adjusted Rand index.

### 1. *Mixture simulation*

#### **Output:**

The desired overlap has been met...

Map of misclassification probabilities:

```
[0][0] : 1.000000 [0][1] : 0.002379 [0][2] : 0.054294 [0][3] : 0.002258
[1][0] : 0.001863 [1][1] : 1.000000 [1][2] : 0.000370 [1][3] : 0.000488
[2][0] : 0.045706 [2][1] : 0.000376 [2][2] : 1.000000 [2][3] : 0.004684
[3][0] : 0.002600 [3][1] : 0.000516 [3][2] : 0.002763 [3][3] : 1.000000
Average Overlap:  0.019716
Maximum Overlap:  0.100000 (components:  0 and 2)
Generalized Overlap:  0.033352
```

Mixture parameters:

Pi:

0.250000 0.250000 0.250000 0.250000

Mu:

```
[0] : 0.445194 0.033992 0.911678
[1] : 0.385350 0.874495 0.941413
[2] : 0.017998 0.104340 0.991776
[3] : 0.397707 0.414299 0.455829
```

Sigma:

```
[0] :
0.023173 0.000842 0.002376
0.000842 0.037617 0.015898
0.002376 0.015898 0.017901
[1] :
0.013202 0.001560 - 0.002515
0.001560 0.015541 - 0.000325
-0.002515 - 0.000325 0.007927
[2] :
0.017999 0.003314 - 0.004907
0.003314 0.014814 0.007064
-0.004907 0.007064 0.056890
[3] :
0.022581 - 0.016514 - 0.000273
-0.016514 0.030775 - 0.000842
-0.000273 - 0.000842 0.008706
```

Dataset with cluster sizes Nk = 31 28 23 18 has been generated...

### 2. *Running R CMD BATCH PROC/Mclust.R*

Runs the program `Mclust.R` from the directory `PROC`. The estimated classification is written into the default file `DATA/idEst.dat`.

### 3. *Running AdjRand*

```
./AdjRand -n100
```

This command calculates the adjusted Rand index and saves its value to `AR.dat` in the directory `DATA`.

#### **Output:**

```
ADJUSTED RAND VALUES:  
Ktrue = 4 Kest = 3 : 0.634519
```

### 4.7.3 Command-Line Interface: An alternative approach

An alternative command-line approach is to use the command:

```
./CARP -m0.1 -p3 -G4 -n100 -k1 -K7 -c -A"R CMD BATCH --no-save \  
--no-restore '--args 3 1 7 100 1 DATA/x.dat DATA/idEst.dat' \  
PROC/runMclust.R"
```

Here, parameter values are not provided in the file being specified in the command line and passed to `PROC/runMclust.R`. The output file is as in Example 4.7.2.

## 4.8 Example: Using external clustering metrics - the R function BHI

**Note:** This example requires the R package `clValid` to be installed.

### 4.8.1 Expert Mode for the Advanced User

When it is needed to use metrics measuring clustering performance other than the adjusted Rand index, the option `-E` can be used. This example illustrates the use of a R function `BHI` from the package `clValid` that calculates a biological homogeneity index. The same approach as in previous Examples 4.6.1 and 4.7.1 can be employed. The C program `BHI.c` collects the parameters and passes them to the R program `runBHI.R`. Both programs are included into `CARP`.

The following command again simulates a three-dimensional mixture with 4 components, generates a dataset of size 100 from it. Next, the program `Kmeans` is run to get the classification vector, and `BHI` is invoked for assessing clustering performance.

```
./CARP -m0.2 -p3 -G4 -n100 -AKmeans -EBHI
```

### 1. *Mixture simulation*

#### **Output:**

Map of misclassification probabilities:

```
[0][0] : 1.000000 [0][1] : 0.039084 [0][2] : 0.003227 [0][3] : 0.017701
[1][0] : 0.055826 [1][1] : 1.000000 [1][2] : 0.048964 [1][3] : 0.094561
[2][0] : 0.001574 [2][1] : 0.021713 [2][2] : 1.000000 [2][3] : 0.046037
[3][0] : 0.018637 [3][1] : 0.105439 [3][2] : 0.077478 [3][3] : 1.000000
```

Average Overlap: 0.088373

Maximum Overlap: 0.200000 (components: 1 and 3)

Generalized Overlap: 0.094958

### 2. *Running Kmeans*

```
./Kmeans -p3 -n100 -K4
```

#### **Output:**

OBTAINED CLASSIFICATIONS:

```
3 3 3 3 3 3 3 0 3 3 3 3 3 3 3 2 3 3 3 0 3 2 2 0 2 2 2 2 2 0 2 2 0 2 1
2 2 3 2 2 2 2 1 2 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0
0 1 1 1 1 1 1 0 0 2 1 1 0 1 0 1 1 1 0 2 1 2 1 1 1 2
```

The estimated classification is written into the default file DATA/idEst.dat.

### 3. *Running BHI*

```
./BHI -n100
```

This command calculates the BHI value and saves it to BHI.dat in the directory DATA.

#### **Output:**

BHI VALUES:

0.689335

## 4.8.2 Command-Line interface for the Less-Experienced User

This is an example on how a user can solve a problem described in Example 4.8.1 by means of the command-line interface. The code of BHI.R is provided below.

```
BHI.call <- function(n, nf, IDtrue, IDest, OutName){
```

```

A <- rep(NA, nf)

for (i in 1:nf){

  id.true <- scan(IDtrue, nlines = 1, skip = i - 1)
  id.est <- scan(IDest, nlines = 1, skip = i - 1) + 1

  K <- max(id.true) + 1

  Q <- NULL

  for (k in 1:K){
    Q <- cbind(Q, id.true == k - 1)
  }
  A[i] <- BHI(id.est, Q)

}

write(A, file = OutName, ncol = 1)

}

n <- 100
nf <- 1
IDtrue <- "DATA/idTrue.dat"
IDest <- "DATA/idEst.dat"
OutName <- "DATA/Diag.dat"

if (!require(clValid)) stop("Package clValid is not installed...")

if (inherits(try(BHI.call(n, nf, IDtrue, IDest, OutName)), what = "try-error"))
stop("Problem with BHI.call; check the parameters...")

```

The following command has to be run.

```
./CARP -m0.2 -p3 -G4 -n100 -C -AKmeans -E"R CMD BATCH PROC/BHI.R"
```

Here, a three-dimensional mixture with 4 components will be generated. The desired maximum overlap is 0.2. 100 observations have to be simulated and the clustering program `Kmeans` will be run. The option `-C` implies that the command line interface has to be employed for the third stage. The results should be analyzed by the program `PROC/BHI.R`.

#### 1. *Mixture simulation*

## Output:

The desired overlap has been met...

Map of misclassification probabilities:

```
[0][0] : 1.000000 [0][1] : 0.078396 [0][2] : 0.007818 [0][3] : 0.011741
[1][0] : 0.040082 [1][1] : 1.000000 [1][2] : 0.088941 [1][3] : 0.022675
[2][0] : 0.007311 [2][1] : 0.111059 [2][2] : 1.000000 [2][3] : 0.059550
[3][0] : 0.020479 [3][1] : 0.019435 [3][2] : 0.039872 [3][3] : 1.000000
```

Average Overlap: 0.084560

Maximum Overlap: 0.200000 (components: 1 and 2)

Generalized Overlap: 0.108530

Mixture parameters:

Pi:

0.250000 0.250000 0.250000 0.250000

Mu:

[0] : 0.000854 0.369287 0.445213

[1] : 0.436049 0.211457 0.099461

[2] : 0.864326 0.371878 0.070712

[3] : 0.579693 0.667698 0.512722

Sigma:

[0] :

0.038635 - 0.016309 - 0.016790

-0.016309 0.023665 0.004287

-0.016790 0.004287 0.035961

[1] :

0.030158 0.014129 - 0.005132

0.014129 0.040973 - 0.008547

-0.005132 - 0.008547 0.017845

[2] :

0.054286 0.008621 0.031935

0.008621 0.028930 0.021788

0.031935 0.021788 0.085234

[3] :

0.050044 0.003364 0.006521

0.003364 0.014721 - 0.009202

0.006521 - 0.009202 0.061370

Dataset with cluster sizes Nk = 25 32 26 17 has been generated...

## 2. *Running* Kmeans

```
./Kmeans -p3 -n100 -K4
```

## Output:

OBTAINED CLASSIFICATIONS:

```
3 2 2 3 3 3 3 3 3 3 3 3 2 3 3 3 3 3 3 3 3 3 2 0 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2 2 0 1 2 2 1 2 2 1 2 0 1 2 0 1 0 1 1 1 1 1 1 2 0 1 1 2
1 2 1 1 1 1 1 1 2 0 0 0 0 0 0 0 0 0 0 0 3 0 0 0 0 0 0 0 0 0
```

The estimated classification is written into the default file `DATA/idEst.dat`.

### 3. *Running AdjRand*

```
R CMD BATCH PROC/BHI.R
```

Runs the program BHI from the directory PROC. The BHI value is calculated and saved into `BHI.dat` in the directory DATA.

#### 4.8.3 Command-Line Interface: An alternative approach

An alternative approach similar to the one provided above in Section 4.8.2 is to employ the command

```
./CARP -m0.2 -p3 -G4 -n100 -AKmeans -C -E"R CMD BATCH --no-save \
--no-restore '--args 100 1 DATA/idTrue.dat DATA/idEst.dat \
DATA/Diag.dat' PROC/runBHI.R"
```

Here, parameter values are not provided in the file being specified in the command line and passed to `PROC/runBHI.R`. The output file is as in Example 4.8.2.

## 4.9 Example: Using external clustering metrics - Diag using the corrected R function `classAgreement`

**Note:** This example requires the R packages `e1071` and `combinat` to be installed.

Our second example on using external metrics deals with another R function `classAgreement`. This function returns several characteristics; one of them – the percentage of correctly classified data points – is employed here. Note that the original function `classAgreement` in the package `e1071` does not provide the correct percentage when the labels are switched. We correct the function by considering all permutations of labels and choosing the largest value produced. Of course, our function might be slow when the number of clusters is large.

### 4.9.1 Expert Interface for the Advanced User

As in Example 4.8, 100 3-dimensional observations are generated from a four-component mixture simulated at the first stage. Then, `Diag` is employed for assessing clustering performance.



```
./CARP -m0.2 -p3 -G4 -n100 -AKmeans -EDiag
```

1. *Mixture simulation*

**Output:**

The desired overlap has been met...

Map of misclassification probabilities:

```
[0][0] : 1.000000 [0][1] : 0.000024 [0][2] : 0.000148 [0][3] : 0.111231
[1][0] : 0.000011 [1][1] : 1.000000 [1][2] : 0.000010 [1][3] : 0.000063
[2][0] : 0.000084 [2][1] : 0.000011 [2][2] : 1.000000 [2][3] : 0.000001
[3][0] : 0.088769 [3][1] : 0.000162 [3][2] : 0.000001 [3][3] : 1.000000
```

Average Overlap: 0.033419

Maximum Overlap: 0.200000 (components: 0 and 3)

Generalized Overlap: 0.066672

Mixture parameters:

Pi:

0.250000 0.250000 0.250000 0.250000

Mu:

```
[0] : 0.763585 0.320591 0.115812
[1] : 0.954638 0.672605 0.568749
[2] : 0.311158 0.859337 0.563586
[3] : 0.979996 0.379821 0.102353
```

Sigma:

```
[0]:
0.013616 0.001818 - 0.011630
0.001818 0.016387 - 0.004770
-0.011630 - 0.004770 0.025950
[1]:
0.006110 - 0.002199 0.001077
-0.002199 0.003766 - 0.000613
0.001077 - 0.000613 0.001719
[2]:
0.008628 - 0.005254 0.005142
-0.005254 0.015077 - 0.007041
0.005142 - 0.007041 0.011054
[3]:
0.007987 0.003230 - 0.006669
0.003230 0.011952 - 0.004099
-0.006669 - 0.004099 0.019565
```

Dataset with cluster sizes  $N_k = 23 \ 26 \ 25 \ 26$  has been generated...

## 2. Running Kmeans

```
./Kmeans -p3 -n100 -K4
```

Output:

OBTAINED CLASSIFICATIONS:

3 0 3 0 0 3 0 0 0 3 0 0 3 0 0 3 0 0 3 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1  
1 1 1 1 1 1 1 1 1 1 1 1 2  
3 3 3 3 3 0 3 3 0 3 3 3 3 0 3 3 3 3 3 3 3 0 3 3 0 0

The estimated classification is written into the default file `DATA/idEst.dat`.

### 3. *Running* Diag

```
./Diag -n100
```

This calculates the proportion of correctly classified observations and saves it into DATA/Diag.dat.

Output:

DIAG VALUES:

0.850000

### 4.9.2 Command-Line interface for the Less-Experienced User

In our next illustration, we repeat Example 4.9.1 providing a command line interface specified with the option `-C`. This way, the user avoids writing and running an executable file, instead calling an R program `Diag.R` directly. The program is given in the directory `PROC`. Its code is provided below.

```
Diag.call <- function(n, nf, IDtrue, IDest, OutName){
```

```
A <- rep(NA, nf)
```

```
for (i in 1:nf){
```

```
id.true <- scan(IDtrue, nlines = 1, skip = i - 1)
```

```
Nk <- as.vector(table(id.true))
```

```
id.est <- scan(IDest, nlines = 1, skip = i - 1)
```

```
K <- max(id.true) + 1
```

```
D <- permn(K)
```

```
A[i] <- -Inf
```

```

for (j in 1:length(D)){
AA <- classAgreement(table(rep(D[[j]], Nk), id.est))$diag
if (AA > A[i]) A[i] <- AA

}
}

write(A, file = OutName, ncol = 1)

}

n <- 100
nf <- 1
IDtrue <- "DATA/idTrue.dat"
IDest <- "DATA/idEst.dat"
OutName <- "DATA/Diag.dat"

if (!require(e1071)) stop("Package e1071 is not installed...")
if (!require(combinat)) stop("Package combinat is not installed...")

if (inherits(try(Diag.call(n, nf, IDtrue, IDest, OutName)), what = "try-error"))
stop("Problem with Diag.call; check the parameters...")

```

Consider the following command.

```
./CARP -m0.2 -p3 -G4 -n100 -AKmeans -C -E"R CMD BATCH PROC/Diag.R"
```

### 1. *Mixture simulation*

#### **Output:**

```

The desired overlap has been met...
Map of misclassification probabilities:
[0][0] : 1.000000 [0][1] : 0.004489 [0][2] : 0.024430 [0][3] : 0.138140
[1][0] : 0.004261 [1][1] : 1.000000 [1][2] : 0.039002 [1][3] : 0.017623
[2][0] : 0.024247 [2][1] : 0.041589 [2][2] : 1.000000 [2][3] : 0.028558
[3][0] : 0.061860 [3][1] : 0.020284 [3][2] : 0.022576 [3][3] : 1.000000
Average Overlap:  0.071176
Maximum Overlap:  0.200000 (components:  0 and 3)
Generalized Overlap:  0.087782

Mixture parameters:
Pi:

```

0.250000 0.250000 0.250000 0.250000

**Mu:**

[0] : 0.969456 0.529585 0.750914

[1] : 0.125748 0.026435 0.180617

[2] : 0.244874 0.621881 0.641587

[3] : 0.947454 0.164489 0.354439

**Sigma:**

[0]:

0.033393 0.003589 0.007757

0.003589 0.055336 0.027874

0.007757 0.027874 0.049052

[1]:

0.042855 - 0.015837 0.030809

-0.015837 0.072075 - 0.053728

0.030809 - 0.053728 0.139985

[2]:

0.057084 - 0.015064 - 0.048054

-0.015064 0.048314 0.034099

-0.048054 0.034099 0.122098

[3]:

0.054242 0.008795 0.012863

0.008795 0.038632 0.007440

0.012863 0.007440 0.016785

Dataset with cluster sizes  $N_k = 19 \ 23 \ 29 \ 29$  has been generated...

## 2. *Running Kmeans*

```
./Kmeans -p3 -n100 -K4
```

**Output:**

OBTAINED CLASSIFICATIONS:

```
0 0 2 2 2 2 2 2 2 2 2 0 2 2 2 2 2 0 0 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 3 3 2 3 1 2 2 3 3 3 2 1 3 2 2 1 1 1 3 1 2 2 3 2 0 2 1 2 2 0 0 0
0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1
```

The estimated classification is written into the default file DATA/idEst.dat.

## 3. *Running PROC/Diag*

```
R CMD BATCH PROC/Diag.R
```

This calculates the proportion of correctly classified observations and saves it into DATA/Diag.dat.

### 4.9.3 Command-Line Interface: An alternative approach

Similar to Example 4.8.3, here the command line interface is employed as indicated by the option `-C`. Thus, the user specifies the parameters in the program `PROC/Diag.R` and avoids writing or modifying a wrapping program to pass the parameters. Alternatively, the user can employ the direct command:

```
./CARP -m0.2 -p3 -G4 -n100 -AKmeans -C -E"R CMD BATCH --no-save \  
--no-restore '--args 100 1 DATA/idTrue.dat DATA/idEst.dat \  
DATA/Diag.dat' PROC/runDiag.R"
```

This command solves the same problem as 4.9.2. However, it does it by passing parameters from the command line rather than specifying them in the file directly. The output file is as in Example 4.9.1 or Example 4.9.2.

### 4.10 Example: Simulating non-Gaussian mixtures

This example illustrates the use of the inverse Box-Cox transformation. Consider the following command.

```
./CARP -m0.01 -p2 -G3 -n500 -Vlambda.dat
```

The transformation parameters are read from the file `lambda.dat` specified by the option `-V`. A bivariate mixture with three components satisfying the maximum overlap 0.01 is generated and a dataset consisting of 500 observations is simulated. As the option `-A` is not specified, no clustering analysis will be performed.

#### 1. *Mixture simulation*

##### **Output:**

```
The desired overlap has been met...  
Map of misclassification probabilities:  
[0][0] : 1.000000 [0][1] : 0.000114 [0][2] : 0.000019  
[1][0] : 0.000085 [1][1] : 1.000000 [1][2] : 0.001803  
[2][0] : 0.000059 [2][1] : 0.008197 [2][2] : 1.000000  
Average Overlap: 0.003425  
Maximum Overlap: 0.010000 (components: 1 and 2)  
Generalized Overlap: 0.005000
```

```
Mixture parameters:  
Pi:  
0.333333 0.333333 0.333333  
Mu:  
[0] : 0.946088 0.691365
```

```

[1] : 0.702896 0.414001
[2] : 0.364032 0.019451
Sigma:
[0] :
0.002423 - 0.001696
-0.001696 0.010649
[1] :
0.003175 - 0.002855
-0.002855 0.006636
[2] :
0.016639 0.010713
0.010713 0.014938
Dataset with cluster sizes Nk = 159 165 176 has been generated...

```

The dataset is simulated and saved in `DATA/x.dat`. Also random transformation coefficient will be saved into `DATA/lambda.dat`. Figure 1 illustrates one dataset simulated by CARP (plot a) and three datasets (plots b-d) obtained from it with an inverse Box-Cox transformation. It can be easily seen that the components have various nonnormal shapes.

## 4.11 Example: Simulating noise coordinates

This example demonstrates how to generate noise coordinates. It also illustrates how noise variables can degrade the performance of clustering algorithms. Consider the following command.

```
./CARP -m0.01 -p1 -G3 -n100 -Ahierclust -w1
```

One-dimensional mixture with 3 components satisfying the maximum overlap 0.01 is simulated and 100 observations are generated from it. Next, one noise variable specified by the option `-w` is added to the dataset. Then, `hierclust` is run and adjusted Rand index is calculated.

### 1. Mixture simulation

The dataset simulated by this command is provided in Figure 2.

#### Output:

```

The desired overlap has been met...
Map of misclassification probabilities:
[0][0] : 1.000000 [0][1] : 0.000000 [0][2] : 0.001859
[1][0] : 0.000000 [1][1] : 1.000000 [1][2] : 0.000000
[2][0] : 0.008141 [2][1] : 0.000000 [2][2] : 1.000000

```

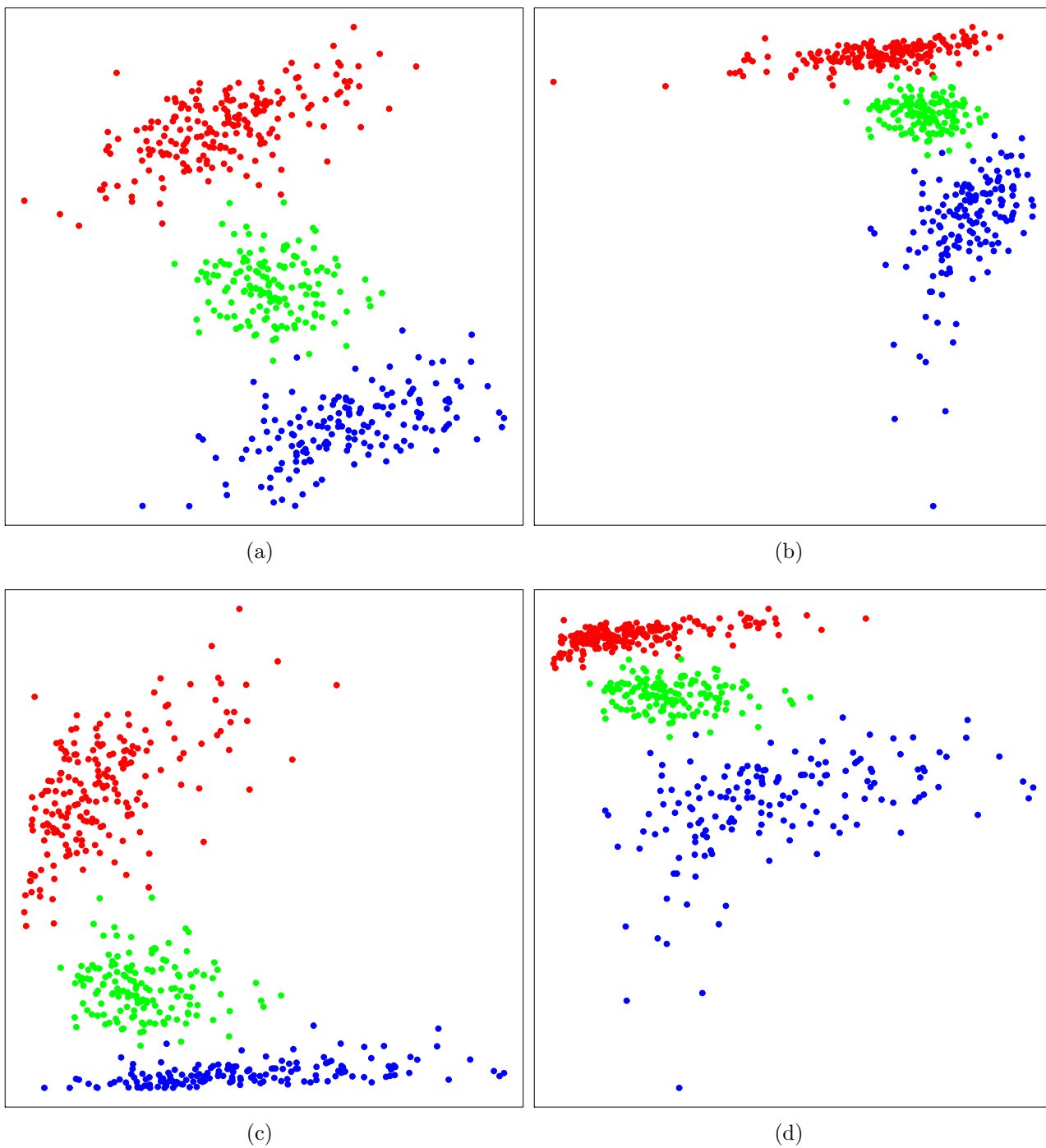


Figure 1: A simulated dataset (a) and three datasets obtained by an inverse Box-Cox transformation (b-d).

```
Average Overlap: 0.003333
Maximum Overlap: 0.010000 (components: 0 and 2)
Generalized Overlap: 0.005000
```

Mixture parameters:

Pi:

0.333333 0.333333 0.333333

Mu:

[0] : 0.808381

[1] : 0.187711

[2] : 0.638410

Sigma:

[0]:

0.000203

[1]:

0.000326

[2]:

0.002866

Dataset with cluster sizes  $N_k = 45 \ 22 \ 33$  has been generated...

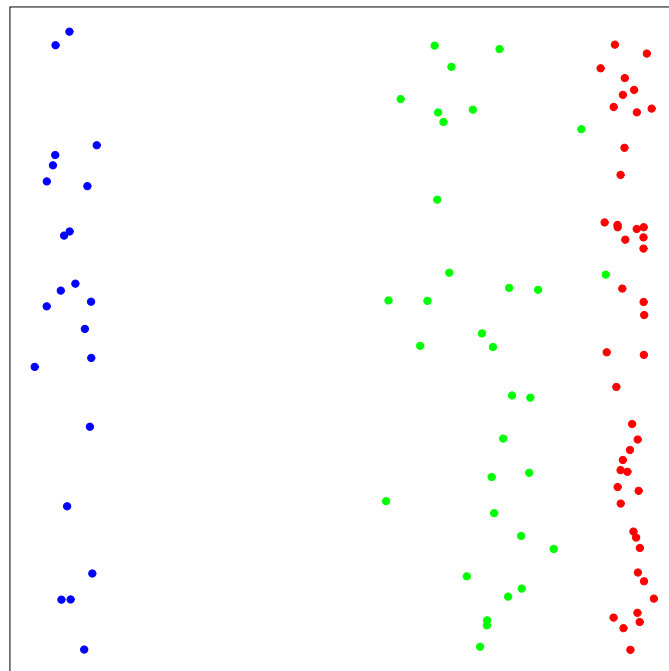


Figure 2: Dataset simulated in Example 4.11.

## 2. *Running hierclust*

```
./hierclust -p2 -n100 -K3
```



### Output:

OBTAINED CLASSIFICATIONS:

```
0 1 1 1 1 1 0 1 0 0 0 1 0 1 0 1 1 0 0 1 0 1 0 0 1 0 1 0 0 0 1 0 1 0 1
0 0 0 0 0 1 1 0 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 0 0 1 1 1 0 0
0 1 1 1 1 0 0 1 1 0 1 1 0 0 0 0 0 0 0 1 0 1 0 1 0 1
```

The estimated classification is written into the default file `DATA/idEst.dat`.

### 3. *Running AdjRand*

```
./AdjRand -n100
```

### Output:

ADJUSTED RAND VALUES:

```
Ktrue = 3 Kest = 3 : 0.326293
```

The generated dataset with noise coordinates is saved into `DATA/x.dat`. The adjusted Rand index is saved into `DATA/AR.dat`. The adjusted Rand index has a very low value indicating that clustering procedure `hierclust` was not able to find reasonable groupings in the simulated dataset despite the low level of original overlap between the components of the mixture.

## 4.12 Example: Simulating datasets with outliers

Our final example illustrates how outlying observations can be generated. Consider the following command.

```
./CARP -m0.01 -p2 -G3 -n100 -Ahierclust -o10
```

A two-dimensional mixture with 3 components is generated and a sample of size 100 is simulated from this mixture. The maximum pairwise overlap is 0.01. The clustering program specified by the option `-A` is `hierclust`. The option `-o` implies that 10 outlying observations have to be added to the main dataset.

### 1. *Mixture simulation*

### Output:

The desired overlap has been met...

Map of misclassification probabilities:

```
[0][0] : 1.000000 [0][1] : 0.006013 [0][2] : 0.000035
[1][0] : 0.003987 [1][1] : 1.000000 [1][2] : 0.000000
[2][0] : 0.000060 [2][1] : 0.000000 [2][2] : 1.000000
```

```

Average Overlap:  0.003365
Maximum Overlap:  0.010000 (components:  0 and 1)
Generalized Overlap:  0.005000

```

Mixture parameters:

Pi:

```
0.333333 0.333333 0.333333
```

Mu:

```
[0] : 0.344086 0.555815
```

```
[1] : 0.435805 0.770761
```

```
[2] : 0.510324 0.213570
```

Sigma:

[0]:

```
0.003079 0.000954
```

```
0.000954 0.001437
```

[1]:

```
0.001797 - 0.001552
```

```
-0.001552 0.003216
```

[2]:

```
0.006978 - 0.002444
```

```
-0.002444 0.002631
```

Dataset with cluster sizes  $N_k = 30 \ 33 \ 37$  has been generated...

10 outliers has been generated...

The simulated dataset is provided in Figure 1. Outliers are given in magenta color.

## 2. *Running hierclust*

```
./hierclust -p2 -n110 -K4
```

An important remark here is that the option `-K` takes value 4 in the above command. This happens because all outlying observations are considered being from another distinct class.

**Output:**

OBTAINED CLASSIFICATIONS:

```

0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 0 0 0 0 0 0 0 0 0 0 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 3 3 1 1 1 3 1 1 1
3 3 1 1 1 1 3 1 1 3 1 1 1 1 1 1 3 1 1 1 1 1 3 1 1 2 2 0 3 3 0 2 3 2 2

```

The estimated classification is written into the default file `DATA/idEst.dat`.

## 3. *Running AdjRand*

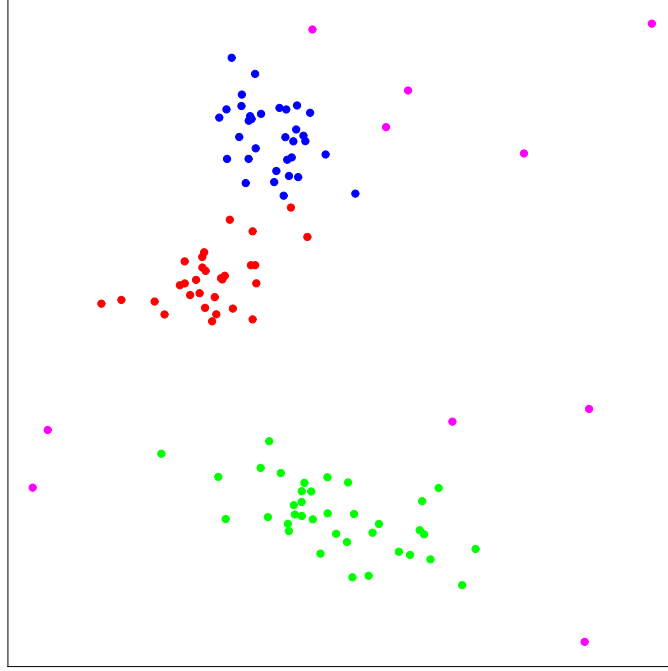


Figure 3: Dataset simulated in Example 4.12.

```
./AdjRand -n110
```

**Output:**

```
ADJUSTED RAND VALUES:
Ktrue = 4 Kest = 4 : 0.937755
```

The generated dataset with noise coordinates is saved into `DATA/x.dat`. The adjusted Rand index is saved into `DATA/AR.dat`.

## 5 An Illustrative Comprehensive Example: Comparing several clustering algorithms

In this chapter, we provide a small example to demonstrate the utility of CARP in order to illustrate how the package can be used to develop clustering algorithms. Our goal is to obtain a comparative study of these algorithms. We evaluate performance of several clustering algorithms each on 25 datasets of size 100 simulated from 5-dimensional mixtures with 7 components and levels of maximum overlap ( $\tilde{\omega}$ ) varying from 0.001 (highly separated) to 0.500 (barely distinguishable). The methods being compared are hierarchical clustering algorithm with Ward's, complete, average, and single linkage,  $k$ -means, and partitioning around medoids algorithm. The comparison is carried out based on the adjusted Rand index.

To run the comparison studies using CARP, the user can proceed as follows.

- *Dataset simulation.* The user runs the following command changing only the level of maximum overlap specified by the option `-m`:  
`./CARP -m0.001 -p5 -G7 -#25 -n100`
- *Hierarchical algorithms.* The user already has the simulated dataset, therefore the simulation stage has to be skipped. Options `-m` and `-b` are therefore omitted. In order to use simulated datasets, the sample size specified by the option `-n` has to be given as a negative number. The program `hierclust` can be employed now. The default linkage in `hierclust` is Ward's linkage. If a user needs to specify a different linkage, the parameter `hlink` has to be changed in `hierclust.c`. Values of `hlink` equal to 1,2,3, and 4 correspond to Ward's, single, complete, and average linkages correspondingly. The following command runs hierarchical clustering using the program `hierclust`.  
`./CARP -p5 -G7 -#25 -n-100 -Ahierclust`
- *k-means algorithm.* A similar command can be provided for running *k*-means algorithm. It will be run on the same datasets used for the evaluation of hierarchical clustering algorithms.  
`./CARP -p5 -G7 -#25 -n-100 -AKmeans`
- *Partitioning around medoids algorithm.* A similar command can be also provided in this case. Again, it runs on the same set of data as before.  
`./CARP -p5 -G7 -#25 -n-100 -APAM`

After these 25 adjusted Rands are obtained for each method, they can be analyzed. These are summarized in Table 1. Here,  $\mathcal{R}_{0.5}$  and  $\mathcal{I}_{\mathcal{R}}$  represent the median of adjusted Rand index and its inter-quartile range respectively. As we can see, hierarchical clustering with Ward's and single linkages are, correspondingly, the best and worst performers in many cases, with *k*-means being second-best.

## 6 Illustration of CARP to evaluate separation on a given dataset: *Iris*

In this section, we briefly illustrate how CARP can be used for the analysis of real life dataset. The famous *Iris* dataset that was already analyzed in Example 4.4 is presented here. Four different clustering methods – hierarchical clustering with Ward's linkage (`hierclust`), *k*-means algorithm (`Kmeans`), expectation-maximization algorithm (`Mclust`), and partitioning around medoids algorithm (`PAM`) – were applied to the dataset. Four different measures assessing performance of clustering were used: adjusted Rand index (`AdjRand`), Rand index (`Rand`), biological homogeneity index (`BHI`), and the proportion of correctly classified observations (`Diag`). The results are presented in Table 2.

The results clearly suggest that the expectation-maximization algorithm realized in the procedure `Mclust` is the best performer. `Kmeans` and `PAM` obtain the same classification vector. `hierclust` does marginally better than the latter two. This example illustrates how CARP can be used for the analysis of real life datasets.

Table 1: Performance of hierarchical algorithms with Ward’s (W), complete (C), average (A), single (S) linkages,  $k$ -means (K), and partitioning around medoids (P) algorithms on clustering 5-dimensional datasets of size 100 with 7 groups.  $\mathcal{R}_{0.5}$  and  $\mathcal{I}_{\mathcal{R}}$  represent the median and inter-quartile range of  $\mathcal{R}$  over 100 replications.

	$\tilde{\omega}$	0.500	0.400	0.300	0.200	0.150	0.100	0.050	0.010	0.005	0.001
W	$\mathcal{R}_{0.5}$	0.082	0.199	0.385	0.582	0.715	0.793	0.931	0.980	1.000	1.000
	$\mathcal{I}_{\mathcal{R}}$	0.061	0.146	0.197	0.300	0.210	0.161	0.105	0.044	0.024	0.000
C	$\mathcal{R}_{0.5}$	0.063	0.164	0.327	0.477	0.596	0.714	0.787	0.956	0.991	1.000
	$\mathcal{I}_{\mathcal{R}}$	0.052	0.125	0.216	0.257	0.231	0.216	0.182	0.145	0.052	0.005
A	$\mathcal{R}_{0.5}$	0.015	0.098	0.227	0.433	0.564	0.689	0.812	0.958	0.990	1.000
	$\mathcal{I}_{\mathcal{R}}$	0.022	0.176	0.242	0.205	0.261	0.179	0.133	0.141	0.111	0.000
S	$\mathcal{R}_{0.5}$	0.002	0.003	0.005	0.017	0.058	0.234	0.490	0.820	0.833	0.897
	$\mathcal{I}_{\mathcal{R}}$	0.007	0.007	0.010	0.202	0.321	0.442	0.449	0.116	0.148	0.170
K	$\mathcal{R}_{0.5}$	0.086	0.230	0.388	0.621	0.671	0.821	0.912	0.978	0.992	1.000
	$\mathcal{I}_{\mathcal{R}}$	0.076	0.149	0.212	0.201	0.212	0.122	0.094	0.029	0.051	0.000
P	$\mathcal{R}_{0.5}$	0.083	0.209	0.341	0.560	0.677	0.811	0.901	0.977	0.982	1.000
	$\mathcal{I}_{\mathcal{R}}$	0.059	0.147	0.237	0.208	0.217	0.192	0.102	0.043	0.027	0.000

Table 2: Performance of various methods (programs `hierclust`, `Kmeans`, `Mclust`, `PAM`) in clustering *Iris* dataset represented by different metrics (programs `AdjRand`, `Rand`, `Diag`, `BHI`).

Method	AdjRand	Rand	Diag	BHI
<code>hierclust</code>	0.731	0.880	0.893	0.860
<code>Kmeans</code>	0.730	0.880	0.893	0.847
<code>Mclust</code>	0.904	0.957	0.967	0.944
<code>PAM</code>	0.730	0.880	0.893	0.847

## 7 Conclusions

The package CARP is a software tool devoted to evaluating performances of finite mixture modeling and clustering algorithms. The underlying technique involves producing Gaussian finite mixture models with pre-specified level of average and maximum pairwise overlap. The adjusted Rand Index is used by default, for assessing the performance of the clustering method under investigation but any other user-specified measure may be used.

## A Appendix: The R package MIXSIM

Here, we present documentation for the related R package MIXSIM, version 1.0-2. The package includes seven functions, `MixSim`, `simdataset`, `overlap`, `pdplot`, `RandIndex`, `VarInf`, and with usage and illustrative examples discussed in the following subsections.

## A.1 MixSim: mixture simulation

The function `MixSim` generates a finite mixture model with Gaussian components according to pre-specified levels of maximum and/or average overlap. The command has the following form:

```
MixSim(BarOmega = NULL, MaxOmega = NULL, K, p, sph = FALSE, hom = FALSE, ecc = 0.90, PiLow = 1.0, int = c(0.0, 1.0), resN = 100, eps = 1e-06, lim = 1e06)
```

The following arguments can be used in `MixSim`:

- `BarOmega` : value of desired average overlap
- `MaxOmega` : value of desired maximum overlap
- `K` : number of components
- `p` : number of dimensions
- `sph` : covariance matrix structure (FALSE - nonspherical, TRUE - spherical)
- `hom` : heterogeneous or homogeneous clusters (FALSE - heterogeneous, TRUE - homogeneous)
- `ecc` : maximum eccentricity
- `PiLow` : value of the smallest mixing proportion (if `PiLow` is not reachable with respect to `K`, equal proportions are taken; `PiLow` = 1.0 implies equal proportions by default)
- `int` : mean vectors are simulated uniformly on a hypercube with sides specified by `int` = (`lower.bound`, `upper.bound`)
- `resN` : maximum number of dataset resimulations
- `eps` : error bound for overlap computation
- `lim` : maximum number of integration terms (Davies, 1980)

If `BarOmega` is not specified, the function generates a mixture solely based on `MaxOmega`; if `MaxOmega` is not specified, the function generates a mixture solely based on `BarOmega`.

The following parameters are returned by `MixSim`:

- `Pi` : vector of mixing proportions
- `Mu` : matrix consisting of components' mean vectors ( $K \times p$ ) `S` : set of components' covariance matrices ( $p \times p \times K$ )
- `OmegaMap` : matrix of misclassification probabilities ( $K \times K$ ); `OmegaMap[i,j]` is the probability that `X` coming from the `i`-th component is classified to the `j`-th component
- `BarOmega` : value of average overlap

- `MaxOmega` : value of maximum overlap
- `rcMax` : row and column numbers for the pair of components producing maximum overlap 'MaxOmega'
- `fail` : flag value; 0 represents successful mixture generation, 1 represents failure

The following examples illustrate the use of the function `MixSim`.

```
# controls average and maximum overlaps
MixSim(BarOmega = 0.05, MaxOmega = 0.15, K = 4, p = 5)

# controls average overlap
MixSim(BarOmega = 0.05, K = 4, p = 5)

# controls maximum overlap
MixSim(MaxOmega = 0.15, K = 4, p = 5)
```

## A.2 `simdataset`: dataset simulation

Function `simdataset` simulates a datasets of the sample size `n` given parameters of finite mixture model with Gaussian components. `simdataset` has the following format:

```
simdataset(n, Pi, Mu, S, n.noise = 0, n.out = 0, alpha = 0.001, max.out = 100000,
int = NULL, lambda = NULL)
```

The following arguments can be used in `simdataset`:

- `n` : sample size
- `Pi` : vector of mixing proportions (length `K`)
- `Mu` : matrix consisting of components' mean vectors ( $K \times p$ )
- `S` : set of components' covariance matrices ( $p \times p \times K$ )
- `n.noise` : number of noise variables
- `n.out` : number of outlying observations
- `n.alpha` : level for simulating outliers
- `max.out` : maximum number of trials to simulate outliers
- `int` : interval for noise and outlier generation
- `lambda` : inverse Box-Cox transformation coefficients

Numbers of observations in components are provided as a realization from a multinomial distribution with probabilities given by mixing proportions.

`simdataset` returns the following parameters:

- `X` : simulated dataset ( $n \times p$ )
- `id` : classification vector (length  $n$ )

The following code illustrates the use of the function `simdataset`.

```
K <- 4
repeat{
  Q <- MixSim(BarOmega = 0.01, MaxOmega = 0.05, K = 4, p = 2)
  if (Q$fail == 0) break
}
A <- simdataset(n = 1000, Pi = Q$Pi, Mu = Q$Mu, S = Q$S)
colors <- c("red", "green", "blue", "brown")
plot(A$X, xlab = "x1", ylab = "x2", type = "n")
for (k in 1:K){
  points(A$X[A$id == k, ], col = colors[k], pch = 19, cex = 0.4)
}
```

### A.3 overlap: overlap calculation

The function `overlap` allows computing misclassification probabilities and pairwise overlaps for finite mixture models with Gaussian components. Overlap is defined as the sum of two misclassification probabilities. `overlap` can be called in the following way:

```
overlap(Pi, Mu, S, eps = 1e-06, lim = 1e06)
```

The following arguments can be accepted by `overlap`:

- `Pi` : vector of mixing proportions (length  $K$ )
- `Mu` : matrix consisting of components' mean vectors ( $K \times p$ )
- `S` : set of components' covariance matrices ( $p \times p \times K$ )
- `eps` : error bound for overlap computation
- `lim` : maximum number of integration terms (Davies, 1980)

The following parameters are returned by `overlap`:

- `OmegaMap`: matrix of misclassification probabilities ( $K \times K$ ); `OmegaMap[i,j]` is the probability that  $X$  coming from the  $i$ -th component is classified to the  $j$ -th component.
- `BarOmega` : value of average overlap



- `MaxOmega` : value of maximum overlap
- `rcMax` : row and column numbers for the pair of components producing maximum overlap `MaxOmega`

The following example demonstrates the use of `overlap` for calculating misclassification probabilities and overlaps for the dataset *Iris*.

```
data(iris)

p <- dim(iris)[2] - 1
n <- dim(iris)[1]
K <- 3

id <- as.numeric(iris[,5])
Pi <- NULL
Mu <- NULL
S <- array(rep(0, p * p * K), c(p, p, K))

# estimate mixture parameters
for (k in 1:K){
  Pi <- c(Pi, sum(id == k) / n)
  Mu <- rbind(Mu, apply(iris[id == k,-5], 2, mean))
  S[, ,k] <- var(iris[id == k,-5])
}

overlap(Pi = Pi, Mu = Mu, S = S)
```

The output produced by `overlap` for *Iris* is given below.

```
$OmegaMap
[,1]          [,2]          [,3]
[1,]1.000000e+00 7.201413e-08 0.000000000
[2,]1.158418e-07 1.000000e+00 0.02302315
[3,]0.000000e+00 2.629446e-02 1.000000000

$BarOmega
[1]0.01643926

$MaxOmega
[1]0.0493176

$rcMax
[1]2 3
```

## A.4 pdplot: parallel distribution plot

`pdplot` is responsible for constructing a parallel distribution plot for a Gaussian finite mixture model. The function has the form given by:

```
pdplot(Pi, Mu, S, file = NULL, Nx = 5, Ny = 5, MaxInt = 1,
marg = c(2,1,1,1))
```

The following arguments can be provided for `pdplot`:

- `Pi` : vector of mixing proportions
- `Mu` : matrix consisting of components' mean vectors ( $K \times p$ )
- `S` : set of components' covariance matrices ( $p \times p \times K$ )
- `file` : name of a .pdf-file
- `Nx` : number of color levels for smoothing along x-axis
- `Ny` : number of color levels for smoothing along y-axis
- `MaxInt` : maximum color intensity
- `marg` : plot margins

The following example provides an illustration to the function `pdplot`. The parallel distribution plot is constructed for the popular *Iris* dataset. Figure 4 illustrates the example.

```
data(iris)

K <- 3
p <- dim(iris)[2] - 1
n <- dim(iris)[1]
id <- as.numeric(iris[,5])
Pi <- NULL
Mu <- NULL
S <- array(rep(0, p * p * K), c(p, p, K))

# estimate mixture parameters
for (k in 1:K){
  Pi <- c(Pi, sum(id == k) / n)
  Mu <- rbind(Mu, apply(iris[id == k,-5], 2, mean))
  S[, ,k] <- var(iris[id == k,-5])
}

pdplot(Pi = Pi, Mu = Mu, S = S)
```

## A.5 RandIndex: index calculation

The function `RandIndex` calculates Rand (1971), adjusted Rand (Hubert and Arabie, 1985), Fowlkes and Mallows (1983) indices, and Mirkin (1996) metric. More information on these and other indices can be found in Meilă (2006). The function has the form given by:

```
RandIndex(id1, id2)
```

The following arguments can be provided for `RandIndex`:

- `id1` : first partitioning vector
- `id2` : second partitioning vector

The following parameters are returned by `RandIndex`:

- `R`: Rand index
- `AR` : adjusted Rand index
- `F` : Fowlkes and Mallows index
- `M` : Mirkin metric

The following example illustrates the utility of the function.

```
id1 <- c(rep(1, 50), rep(2,100))  
id2 <- rep(1:3, each = 50)  
RandIndex(id1, id2)
```

The output produced by `RandIndex` is given below.

```
$R  
[1]0.7762864
```

```
$AR  
[1]0.5681159
```

```
$F  
[1]0.7714543
```

```
$M  
[1]5000
```

## A.6 VarInf: variation of information calculation

The function `VarInf` calculates the variation of information according to Meilă (2006). The index is equal to 0 if the classification vectors are identical.

```
VarInf(id1, id2)
```

The following arguments can be provided for `VarInf`:

- `id1` : first partitioning vector
- `id2` : second partitioning vector

The following example illustrates the function.

```
id1 <- c(rep(1, 50), rep(2,100))  
id2 <- rep(1:3, each = 50)  
VarInf(id1, id2)
```

The output produced by `VarInf` is provided below.

```
[1]0.4620981
```

## A.7 VarInf: variation of information calculation

The function `ClassProp` calculates the agreement proportion for two classification vectors.

```
ClassProp(id1, id2)
```

The following arguments can be provided for `ClassProp`:

- `id1` : first partitioning vector
- `id2` : second partitioning vector

Consider the following example.

```
id1 <- c(rep(1, 50), rep(2,100))  
id2 <- rep(1:3, each = 50)  
ClassProp(id1, id2)
```

The function returns the following output.

```
[1]0.6666667
```

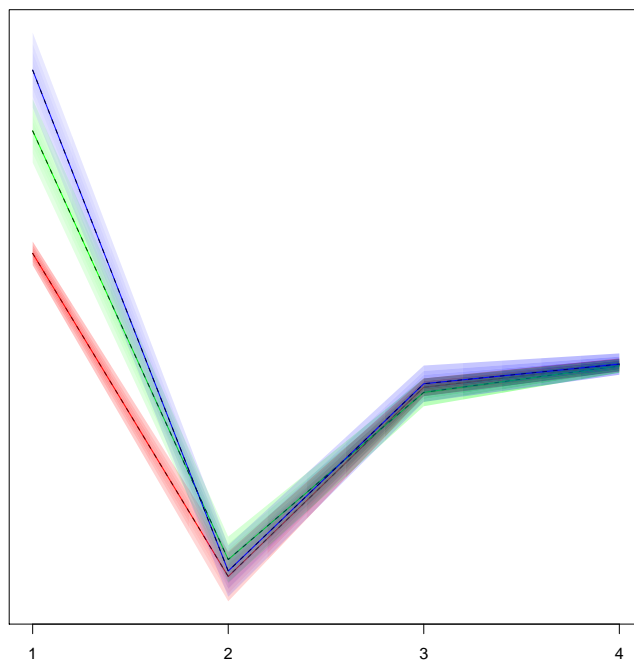


Figure 4: Parallel distribution plot for the *Iris* dataset.

## B Appendix: Geometry of Overlap

To illustrate the geometry of overlap for different measures, mixture distributions with various overlap levels have been simulated. For generalized ( $\tilde{\omega}$ ), maximum ( $\tilde{\omega}$ ), and average ( $\bar{\omega}$ ) overlap, we use the following four values: 0.001, 0.01, 0.1, and 0.2. Corresponding mixtures and sample distributions are provided in Figures 5, 6, and 7 respectively. Figure 8 contains sample distributions for the cases when both maximum and average overlaps are specified at the levels  $(\tilde{\omega} : \bar{\omega}) = (0.001 : 0.003)$ ,  $(0.001 : 0.006)$ ,  $(0.01 : 0.03)$ , and  $(0.01 : 0.06)$ . For each level of overlap measure, three four-component mixture distributions in two dimensions have been simulated. Each row of plots corresponds to a particular level of overlap. As we can see, the complexity of mixtures grows along with the increase in  $\tilde{\omega}$ ,  $\tilde{\omega}$ , and  $\bar{\omega}$ : more mixture components start interacting with each other and pairwise overlaps increase. In the case when  $\tilde{\omega}$  and  $\bar{\omega}$  are both specified, we can control the structure of simulated mixtures better. When the levels of  $\tilde{\omega}$  and  $\bar{\omega}$  are closer to each other, more components interact with each other overlapping moderately. On the contrary, when  $\tilde{\omega}$  and  $\bar{\omega}$  are substantially different, few components will substantially contribute to the overlap while the rest will be well separated.

## References

- E. Anderson. The Irises of the Gaspe peninsula. *Bulletin of the American Iris Society*, 59: 2–5, 1935.
- S. Datta and S. Datta. Methods for evaluating clustering algorithms for gene expression data using a reference set of functional classes. *BMC Bioinformatics*, 7:397, 2006.

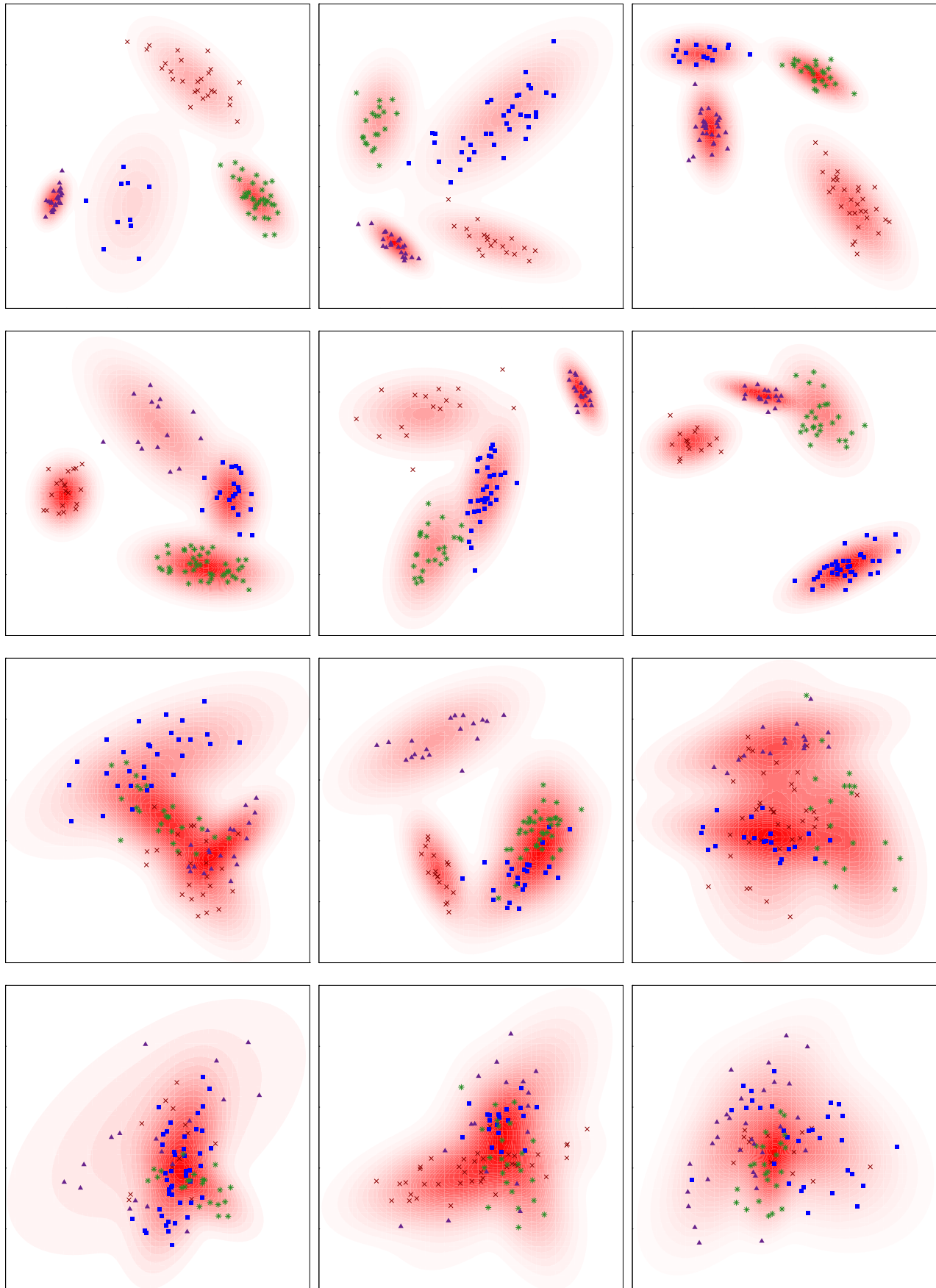


Figure 5: Contour plots of sample four-component mixture distributions in two dimensions for  $\bar{\omega} = 0.001, 0.01, 0.1$ , and  $0.2$  by rows.

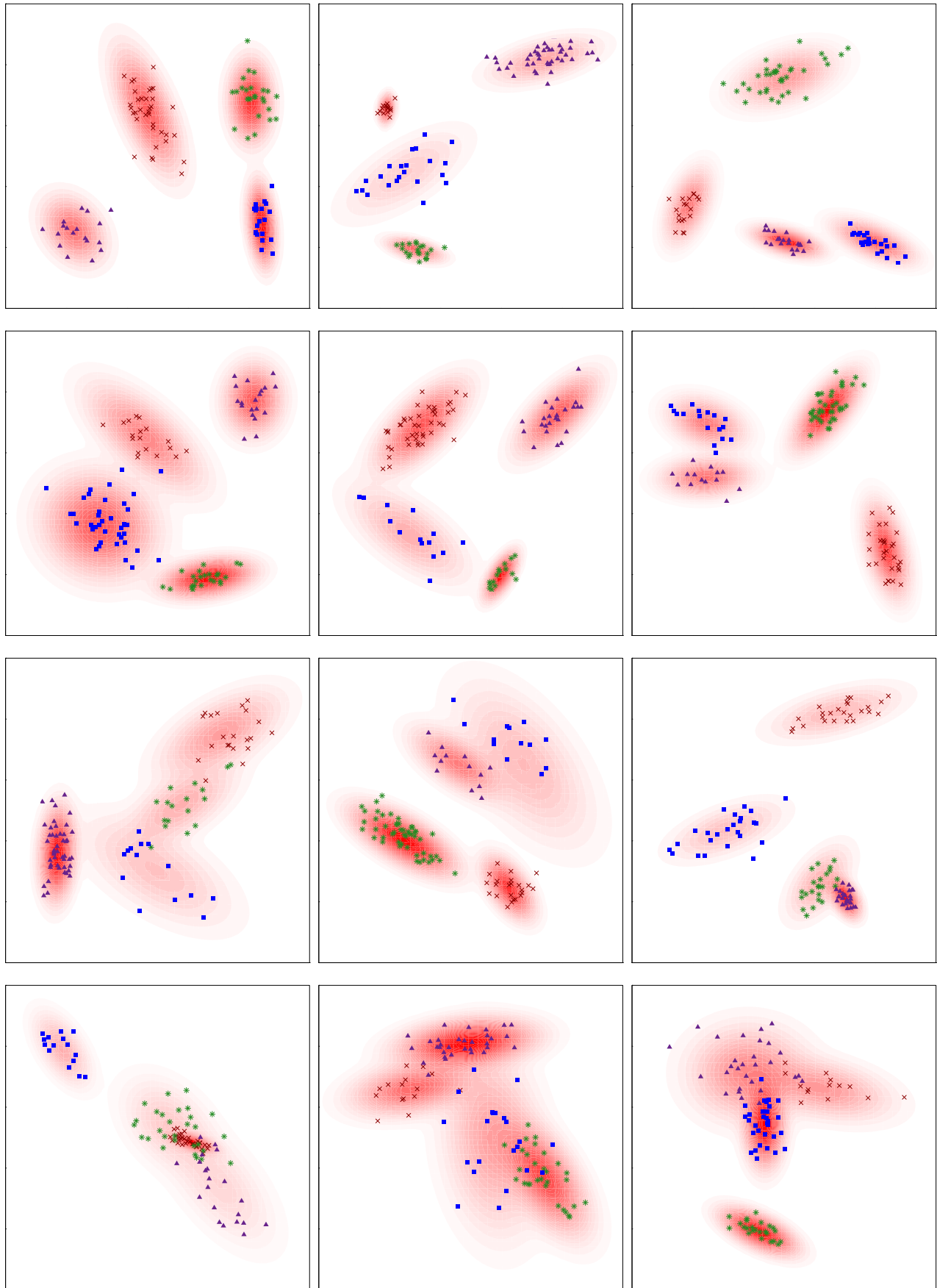


Figure 6: Contour plots of sample four-component mixture distributions in two dimensions for  $\tilde{\omega} = 0.001, 0.01, 0.1$ , and  $0.2$  by rows.

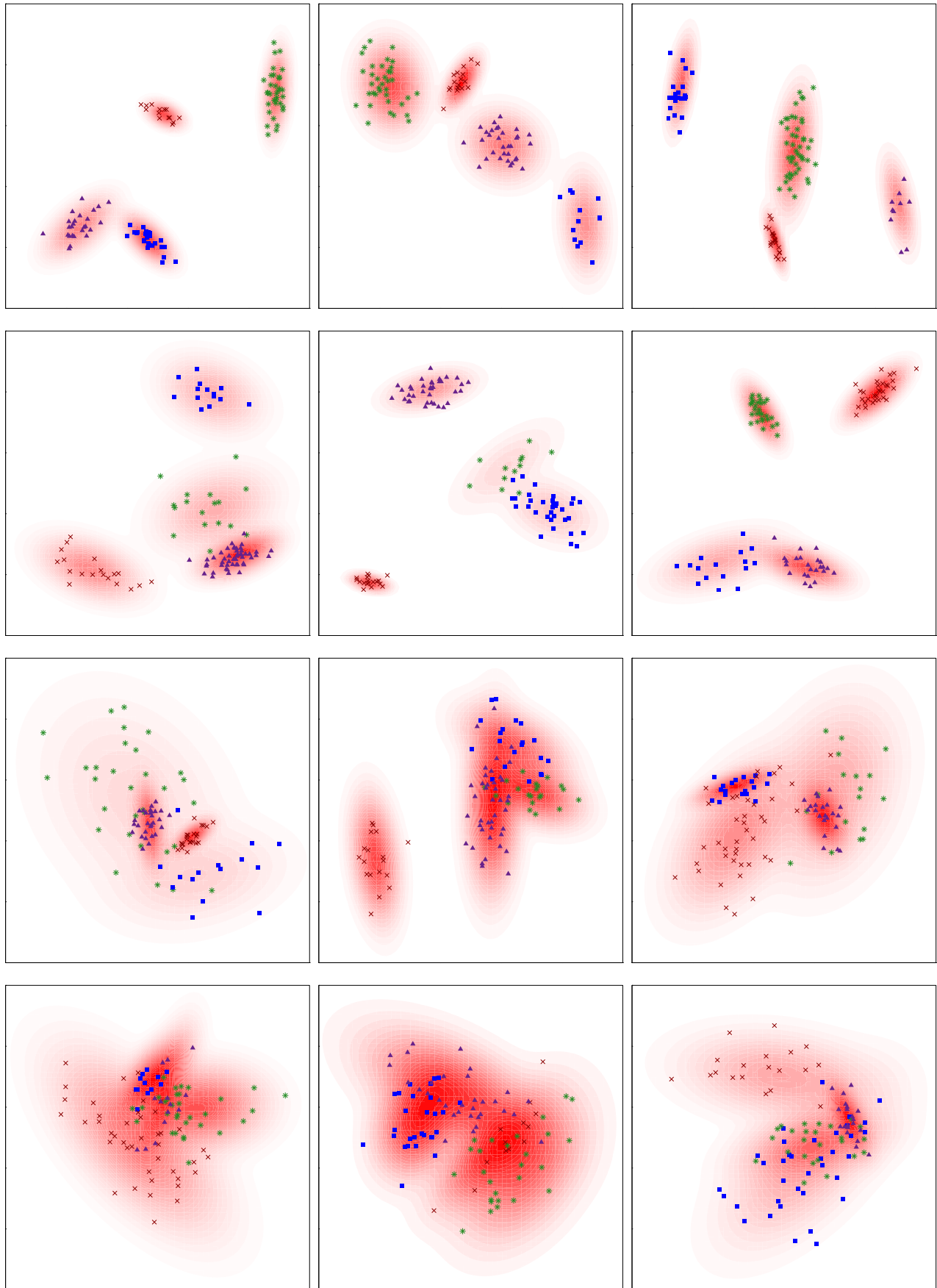


Figure 7: Contour plots of sample four-component mixture distributions in two dimensions for  $\bar{\omega} = 0.001, 0.01, 0.1$ , and  $0.2$  by rows.



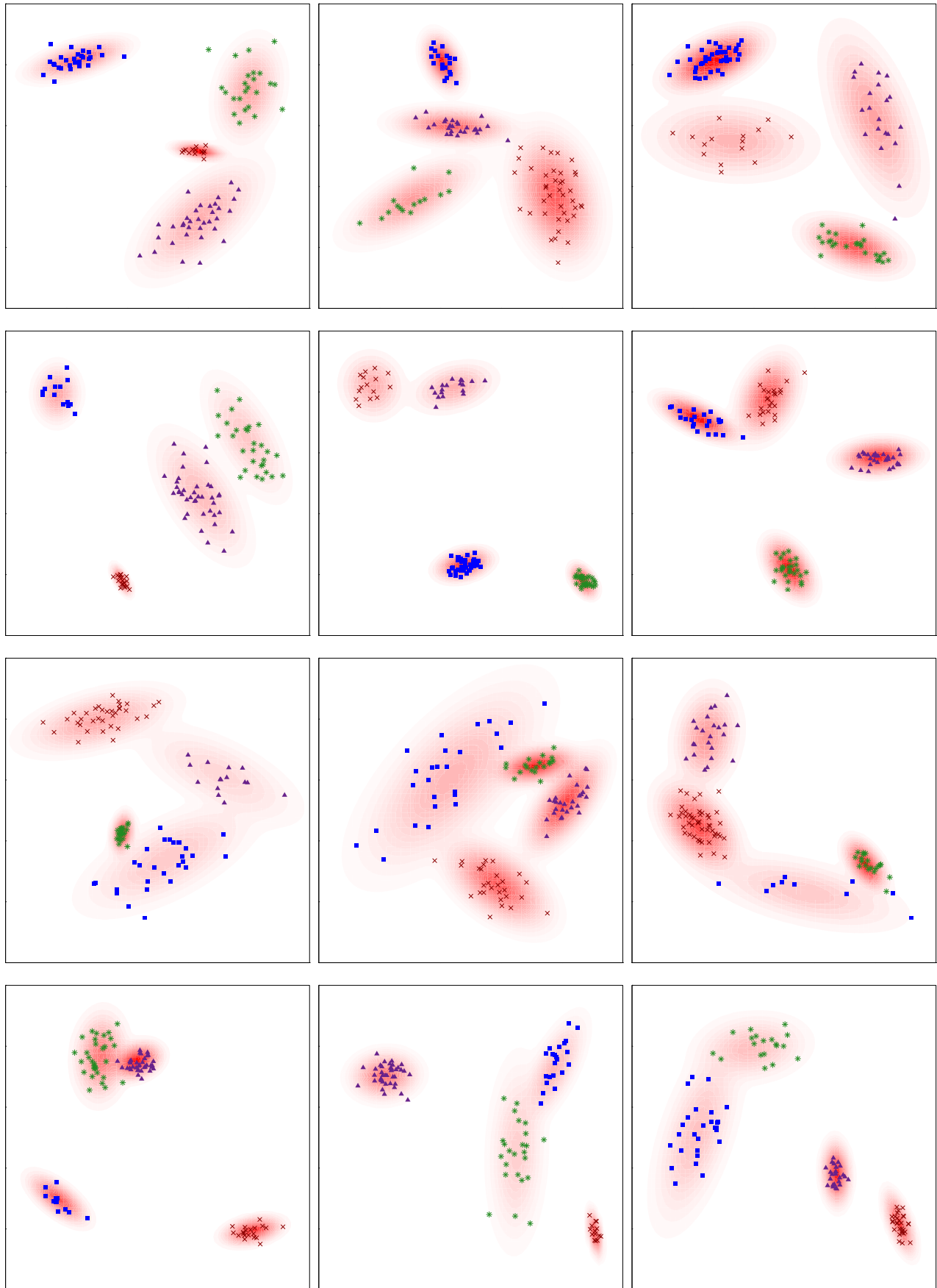


Figure 8: Contour plots of sample four-component mixture distributions in two dimensions for  $(\tilde{\omega} : \bar{\omega}) = (0.001 : 0.003)$ ,  $(0.001 : 0.006)$ ,  $(0.01 : 0.03)$ , and  $(0.01 : 0.06)$  by rows.

- R. Davies. The distribution of a linear combination of  $\chi^2$  random variables. *Applied Statistics*, 29:323–333, 1980.
- A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood for incomplete data via the EM algorithm (with discussion). *Journal of the Royal Statistical Society, Series B*, 39:1–38, 1977.
- R. A. Fisher. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7:179–188, 1936.
- E. Forgy. Cluster analysis of multivariate data: efficiency vs. interpretability of classifications. *Biometrics*, 21:768–780, 1965.
- E. Fowlkes and C. Mallows. A method for comparing two hierarchical clusterings. *Journal of American Statistical Association*, 78:553–569, 1983.
- C. Fraley and A. E. Raftery. MCLUST version 3 for R: Normal mixture modeling and model-based clustering. Technical Report 504, University of Washington, Department of Statistics, Seattle, WA, 2006. 2006.
- J. A. Hartigan and M. A. Wong. A  $k$ -means clustering algorithm. *Applied Statistics*, 28:100–108, 1979.
- L. Hubert and P. Arabie. Comparing partitions. *Journal of Classification*, 2:193–218, 1985.
- L. Kaufman and P. J. Rousseuw. *Finding Groups in Data*. John Wiley and Sons, Inc., New York, 1990.
- J. MacQueen. Some methods for classification and analysis of multivariate observations. *Proceedings of the Fifth Berkeley Symposium*, 1:281–297, 1967.
- R. Maitra. A re-defined and generalized percent-overlap-of-activation measure for studies of fMRI reproducibility and its use in identifying outlier activation maps. *NeuroImage*, 50:124–135, 2010.
- R. Maitra and V. Melnykov. Simulating data to study performance of finite mixture modeling and clustering algorithms. *Journal of Computational and Graphical Statistics*, 19(2):354–376, 2010a. doi: 10.1198/jcgs.2009.08054.
- R. Maitra and V. Melnykov. Assessing significance in finite mixture models. Technical report, Department of Statistics, Iowa State University, 2010b.
- M. Meilă. Comparing clusters – an information based distance. *Journal of Multivariate Analysis*, 98:873–895, 2006.
- B. Mirkin. *Mathematical classification and clustering*. Kluwer Academic Press, Dordrecht, 1996.
- W. M. Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, 66:846–850, 1971.

J. H. Ward. Hierarchical grouping to optimize an objective function. *Journal of the American Statistical Association*, 58:236–244, 1963.