Missingness Aware Gaussian Mixture Models

Zachary R. McCaw

2021-12-13

Contents

Introduction

Overview

This package implements clustering of multivariate normal random vectors with missing elements. Clustering is achieved by fitting a Gaussian Mixture Model (GMM). The parameters are estimated by maximum likelihood, using the Expectation Maximization (EM) algorithm. Our implementation complements existing methods by allowing for missingness in the input data and full covariance matrices. The EM algorithm addresses missingness of both the cluster assignments and the vector components. The output includes the marginal cluster membership probabilities; the mean and covariance of each cluster; the density of each mixture component evaluated at the observations; the posterior probabilities of cluster membership; maximum *a posteriori* cluster assignments; and a completed version of the input data, with missing values imputed to their posterior expectations.

Model

Suppose the data consist of n random vectors in $\mathbb{R}^D$. Each observation Y_i arises from one of K distinct clusters. Associate with each observation a $K \times 1$ vector of indicators $\boldsymbol{Z}_i \in \{0,1\}$, where $Z_{ik} = 1$ if observation i belongs to cluster k, and $Z_{ik} = 0$ otherwise. These cluster membership indicators are latent variables. The marginal probability of membership to cluster k is $\pi_k = P(z_{ik} = 1)$. Conditional on membership to the kth cluster, the observation (example) $\boldsymbol{Y}_i$ follows a multivariate normal distribution, with cluster-specific mean $\boldsymbol{\mu}_k$ and covariance $\boldsymbol{\Sigma}_k$. Overall, the generative model is:

$$\boldsymbol{Z}_i \sim \text{Multinomial}(\pi_1, \cdots, \pi_K)$$
$$\boldsymbol{Y}_i \big| (Z_{ik} = 1) \sim N\big(\mu_k, \Sigma_k\big)$$

Marginally, the observations Y_i follow a GMM, with density:

$$f(\boldsymbol{y}_i) = \sum_{k=1}^{K} f(\boldsymbol{y}_i, z_{ik} = 1) = \sum_{k=1}^{K} f(\boldsymbol{y}_i | z_{ik} = 1) P(z_{ik} = 1) = \sum_{k=1}^{K} f(\boldsymbol{y}_i | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \pi_k.$$

Each element Y_{id} of $\boldsymbol{Y}_i$ is potentially missing at random. Associated with each observation a $D \times 1$ vector of response indicators $\boldsymbol{R}_i$, where $\boldsymbol{R}_{id} = 1$ if element d of observation i is observed, and $R_{id} = 0$ otherwise. Partition each $\boldsymbol{Y}_i$ into its observed $\boldsymbol{Y}_i^{\text{obs}}$ and missing $\boldsymbol{Y}_i^{\text{mis}}$ components. That is, element Y_{id} belongs to $\boldsymbol{Y}_i^{\text{obs}}$ if $R_{id} = 1$, and belongs to $\boldsymbol{Y}_i^{\text{mis}}$ if $R_{id} = 0$. The missingness occurs at random if $(R_{id} \perp Y_{id})|\boldsymbol{Y}_i^{\text{obs}}$. That is, given the observed elements of $\boldsymbol{Y}_i$, whether any remaining element is missing is independent of that element's value.

Maximum likelihood estimates (MLEs) for the parameters of the GMM are obtained using the EM algorithm. During the E-step, both the cluster assignments Z_{ik} and the unobserved components $\boldsymbol{Y}_i^{\text{mis}}$ of $\boldsymbol{Y}_i$ are treated as missing data. Suppose momentarily that all data were observed for observation i. The contribution of subject i to the *complete data* log likelihood would be:

$$\ell_i = \sum_{k=1}^{K} z_{ik} \ln \pi_k - \frac{1}{2} \sum_{k=1}^{K} z_{ik} \ln \det(\boldsymbol{\Sigma}_k) - \frac{1}{2} \sum_{k=1}^{K} z_{ik} (\boldsymbol{y}_i - \boldsymbol{\mu}_k) \boldsymbol{\Sigma}_k^{-1} (\boldsymbol{y}_i - \boldsymbol{\mu}_k).$$

Because the Z_{ik} are not observed, and the $\boldsymbol{Y}_i$ are incompletely observed, the complete data log likelihood cannot be evaluated. Instead, an EM objective is formed by taking the expectation of the complete data log likelihood, conditional on the observed data and the current parameter state:

$$q_i^{(r)} \equiv \mathbb{E}\big(\ell_i | y_i^{\text{obs}}, \vartheta^{(r)}\big) = \sum_{k=1}^{K} \gamma_{ik}^{(r)} \ln \pi_k - \frac{1}{2} \sum_{k=1}^{K} \gamma_{ik}^{(r)} \ln \det(\boldsymbol{\Sigma}_k) - \frac{1}{2} \sum_{k=1}^{K} \operatorname{tr}\big(\boldsymbol{\Sigma}_k^{-1} \boldsymbol{V}_{ik}^{(r)}\big).$$

Here $\boldsymbol{y}_i^{\text{obs}}$ is the observed data for observation i; $\vartheta^{(r)}$ is the current parameter state; $\gamma_{ik}^{(r)}$ is the *responsibility*, defined as $\mathbb{E}(Z_{ik} | y_i^{\text{obs}}, \vartheta^{(r)}]$; and $\boldsymbol{V}_{ik}^{(r)}$ is the *expected residual outer product*, defined as $\mathbb{E}\big\{ Z_{ik} (Y_i - \mu_k)(Y_i - \mu_k)' | y_i^{\text{obs}}, \vartheta^{(r)} \big\}$. In the M-step, updates of the model parameters ϑ are obtained by maximizing the EM objective.

Once the convergence criterion has been achieved, the responsibility, or posterior probability of cluster membership, is calculated as:

$$\gamma_{ik} = \mathbb{P}(Z_{ik} = 1 | \boldsymbol{y}_i^{\text{obs}}) = \frac{f(\boldsymbol{y}_i^{\text{obs}} |\ \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \pi_k}{\sum_{k'=1}^{K} f(\boldsymbol{y}_i^{\text{obs}} |\ \boldsymbol{\mu}_{k'}, \boldsymbol{\Sigma}_{k'}) \pi_{k'}}.$$

The maximum a posteriori classification for $\boldsymbol{y}_i$ is given by:

$$A_i = \arg\max_k\ \gamma_{ik}$$

For observation $\boldsymbol{Y}_i$, posterior expectation of the missing elements $\boldsymbol{Y}_i^{\text{mis}}$, given the observed elements $\boldsymbol{Y}_i^{\text{obs}}$, is:

$$\mathbb{E}(\boldsymbol{Y}_i^{\text{mis}} | y_i^{\text{obs}}) = \sum_{k=1}^{K} \mathbb{E}(\boldsymbol{Y}_i^{\text{mis}} | \boldsymbol{y}_i^{\text{obs}}, z_{ik} = 1) \pi_k.$$

Data Generation

Description

The function `rGMM` simulates observations from a Gaussian Mixture Model. The number of observations is specified by `n`, and the dimension of each observation by `d`. The number of clusters is set using `k`, which defaults to 1. The marginal probabilities of cluster membership are provided as a numeric vector `pi`, which should contain `k` elements. If `pi` is omitted, the clusters are assumed equiprobable. The proportion of elements in the $n \times d$ data matrix that are missing is specified by `miss`, which defaults to zero. Note that when `miss>0`, it is possible for all elements of an observation to be missing. The cluster `means` are provided either as a numeric prototype vector, or as a list of numeric vectors. If a single prototype is provided, that vector is taken as the mean for all clusters. By default, the zero vector is adopted as the prototype. The cluster covariances `covs` are provided as a numeric prototype matrix, or as a list of such matrices. If a single prototype is provided, that matrix is used as the covariance for all clusters. By default, the identity matrix is adopted as the prototype.

Examples

Single Component without Missingness

In this example, `n = 1e3` observations are simulated from a single `k = 1` bivariate normal distribution `d = 2` without missingness. The mean is $\boldsymbol{\mu} = (2, 2)$, and the covariance is an exchangeable correlation structure with off-diagonal $\rho = 0.5$.

```
set.seed(100)
# Single component without missingness.
sigma <- matrix(c(1, 0.5, 0.5, 1), nrow = 2)
data <- rGMM(n = 1e3, d = 2, k = 1, means = c(2, 2), covs = sigma)
```

Single Component with Missingness

In this example, `n = 1e3` observations are simulated from a single `k = 1` trivariate normal distribution `d = 3` with 20% missingness `miss = 0.2`. The mean defaults to the zero vector, and the covariance to the identity matrix.

```
# Single component with missingness.
data <- rGMM(n = 1e3, d = 3, k = 1, miss = 0.2)
```

Two Components without Missingness

In this example, `n = 1e3` observations are simulated from a two-component `k = 2` trivariate normal distribution `d = 3` without missingness. The mean vectors are $\boldsymbol{\mu}_1 = (-2, -2, -2)$ and $\boldsymbol{\mu}_2 = (2, 2, 2)$. The covariance matrices are both exchangeable with off-diagonal $\rho = 0.5$. Because `pi` is omitted, the cluster are equi-probable, i.e. $\pi_1 = \pi_2 = 1/2$.

```
# Two-component mixture without missingness.
mean_list <- list(
  c(-2, -2, -2),
  c(2, 2, 2)
)
sigma <- matrix(
  c(1, 0.5, 0.5,
```

```
    0.5, 1, 0.5,
    0.5, 0.5, 1), nrow = 3)
data <- rGMM(n = 1e3, d = 3, k = 2, means = mean_list, covs = sigma)
```

Four Components with Missingness

In this example, `n = 1e3` observations are simulated from a four-component `k = 4` bivariate normal distribution `d = 2` with 10% missingness `miss = 0.1`. The mean vectors are $\boldsymbol{\mu}_1 = (-2, -2)$, $\boldsymbol{\mu}_2 = (-2, 2)$, $\boldsymbol{\mu}_3 = (2, -2)$ and $\boldsymbol{\mu}_4 = (2, 2)$. The covariance matrices are all $0.5 * I$. The cluster proportions are (35%, 15%, 15%, 35%) for $(\pi_1, \pi_2, \pi_3, \pi_4)$, respectively.

```
# Four-component mixture with missingness.
mean_list <- list(
  c(-2, -2),
  c(-2, 2),
  c(2, -2),
  c(2, 2)
)
sigma <- 0.5 * diag(2)
props <- c(0.35, 0.15, 0.15, 0.35)
data <- rGMM(
  n = 1e3,
  d = 2,
  k = 4,
  pi = props,
  miss = 0.1,
  means = mean_list,
  covs = sigma
)
```

Parameter Estimation

Description

The function `FitGMM` estimates the GMM parameters. The data are expected as a numeric matrix `data`, with observations as rows. The number of mixture components is specified using `k`, which defaults to 1. Initial values for the mean vectors, covariance matrices, and cluster proportions are provided using `init_means`, `init_covs`, and `init_props`, respectively. The initial means `init_means` are provided as a list of vectors, the initial covariances `init_covs` as a list of matrices, and the cluster proportions `init_props` as a numeric vector. Initial means and covariances should be supplied as a lists with `k` components, even if `k = 1`, or if all `k > 1` components are initialized at the same value.

If the `data` contain complete observations, i.e. observations with no missing elements, `FitGMM` will attempt to initialize all model parameters $(\boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\pi})$. However, if the data `data` contain no complete observations, then initial values are required for each of `init_means`, `init_props`, and `init_props`. Supplying initial values may also result in better performance when there are relatively few complete observations.

The arguments `maxit`, `eps`, and `report` control the fitting procedure. `maxit` sets the maximum number of EM iterations to attempt; the default is 10^2. `eps` sets the minimum acceptable improvement in the EM objective function; the default is 10^{-6}. If `report = TRUE`, then fitting progress is displayed.

Examples

Single Component without Missingness

In this example, 10^3 observations are simulated with a single bivariate normal distribution without missingness. The output is an object of class `mvn` containing the estimated mean and covariance, and the log likelihood. In the case of a single component without missingness, the maximum likelihood estimates are available in closed form.

```
# Single component without missingness.
sigma <- matrix(c(1, 0.5, 0.5, 1), nrow = 2)
data <- rGMM(n = 1e3, d = 2, k = 1, means = c(2, 2), covs = sigma)
fit <- FitGMM(data, k = 1)
show(fit)
```

```
## Multivariate Normal Model.
##
## Estimated mean:
##   y1   y2
## 2.02 2.01
##
## Estimated covariance:
##       y1    y2
## y1 1.070 0.532
## y2 0.532 0.999
##
## Final Objective:
## [1] -1750
```

Methods are provided for extracting the mean, covariance, and EM objective value from the `mvn` class. In the case of a single component with complete data, `logLik` in fact returns the log likelihood, but in general the EM objective will differ from the observed data log likelihood.

```r
cat("\nObject class:\n")
class(fit)

cat("\nExtract mean:\n")
mean(fit)

cat("\nExtract covariance:\n")
vcov(fit)

cat("\nExtract EM objective:\n")
suppressWarnings({logLik(fit)})
```

```
##
## Object class:
## [1] "mvn"
## attr(,"package")
## [1] "MGMM"
##
## Extract mean:
##       y1       y2
## 2.024176 2.007562
##
## Extract covariance:
##           y1        y2
## y1 1.0669079 0.5317859
## y2 0.5317859 0.9994667
##
## Extract EM objective:
## [1] -1754.07
```

Single Component with Missingness

In this example, 10^3 observations are simulated from a single trivariate normal distribution with 20% missingness. The output is an object of class `mvn`. In addition to the mean, covariance, and EM objective, `fit@Completed` contains a *completed* version of the input data, with missing values imputed to their posterior expectations. The true mean is the zero vector, and the true covariance is identity. For `fit1` below, the initial mean and covariance are estimated internally using the complete observations. For `fit2` below, the mean and covariance are initialized at the truth. The final value of the EM objective is increased by initializing at the truth.

```r
set.seed(102)

# Single component with missingness.
data <- rGMM(n = 1e3, d = 3, k = 1, miss = 0.2)

cat("Initial parameter values set internally:\n")
fit1 <- FitGMM(data, k = 1)

cat("\nEstimated mean:\n")
mean(fit1)

cat("\nEstimated covariance:\n")
vcov(fit1)
```

```r
cat("\nFinal objective:\n")
logLik(fit1)

cat("\nOriginal data:\n")
head(data)

cat("\nCompleted data:\n")
head(fit1@Completed)

cat("\n\nInitial parameter values set manually:\n")
init_means <- list(c(0, 0, 0))
init_covs <- list(diag(3))
fit2 <- FitGMM(data, k = 1, init_means = init_means, init_covs = init_covs)

cat("\nEstimated mean:\n")
mean(fit2)

cat("\nEstimated covariance:\n")
vcov(fit2)

cat("\nFinal objective:\n")
logLik(fit2)

cat("\nGain in final objective by initializing parameters at the truth:\n")
suppressWarnings({logLik(fit2) - logLik(fit1)})
```

```
## Initial parameter values set internally:
## Objective increment:  1.32
## Objective increment:  0.0425
## Objective increment:  0.00177
## Objective increment:  0.000115
## Objective increment:  1.13e-05
## Objective increment:  1.34e-06
## Objective increment:  1.68e-07
## 6 update(s) performed before reaching tolerance limit.
##
##
## Estimated mean:
##           y1           y2           y3
##  0.007209316  0.053596831 -0.027423821
##
## Estimated covariance:
##             y1          y2         y3
## y1  0.91340883 -0.02162862 0.02771232
## y2 -0.02162862  0.97258027 0.06271627
## y3  0.02771232  0.06271627 0.95024429
##
## Final objective:
## [1] -2793.743
##
## Original data:
##            y1         y2         y3
## 1  0.72468613 -0.1638613 -1.6412945
```

```
## 1  0.67948387  0.7603641  0.6233098
## 1  0.67308290  0.9270161  0.9038616
## 1  0.61192233         NA         NA
## 1  0.06146299  1.0344105 -1.4162869
## 1 -0.74674770  1.3592285  0.9550314
##
## Completed data:
##            y1          y2          y3
## 1  0.72468613 -0.16386131 -1.64129449
## 1  0.67948387  0.76036412  0.62330985
## 1  0.67308290  0.92701612  0.90386163
## 1  0.61192233  0.03927782 -0.00907716
## 1  0.06146299  1.03441052 -1.41628687
## 1 -0.74674770  1.35922852  0.95503144
##
##
## Initial parameter values set manually:
## Objective increment:  8.47
## Objective increment:  0.541
## Objective increment:  0.0465
## Objective increment:  0.00493
## Objective increment:  0.000572
## Objective increment:  6.87e-05
## Objective increment:  8.37e-06
## Objective increment:  1.03e-06
## Objective increment:  1.26e-07
## 8 update(s) performed before reaching tolerance limit.
##
##
## Estimated mean:
##           y1           y2           y3
##  0.007209512  0.053596653 -0.027423582
##
## Estimated covariance:
##             y1          y2         y3
## y1  0.91340891 -0.02163085 0.02771294
## y2 -0.02163085  0.97258018 0.06271506
## y3  0.02771294  0.06271506 0.95024417
##
## Final objective:
## [1] -2793.743
##
## Gain in final objective by initializing parameters at the truth:
## [1] 0.0001295088
```

Two Components without Missingness

In this example, 10^3 observations are simulated from a two-component, trivariate normal distribution without missingness. The output is an object of class `mix` with the following slots: * `@Means` and `@Covariances`: lists of the estimated cluster means and covariances. * `@Density`: the cluster densities evaluated at the observations. * `@Responsibilities`: the posterior membership probabilities for each observation. * `@Assignments`: the maximum a posteriori cluster assignments and assignment entropy. * `@Completed`: a completed version of the input data is returned, with missing values replaced by their posterior expectations

```r
# Two componets without missingness
mean_list <- list(
  c(-2, -2, -2),
  c(2, 2, 2)
)
cov <- matrix(
  c(1, 0.5, 0.5,
    0.5, 1, 0.5,
    0.5, 0.5, 1), nrow = 3
)

data <- rGMM(n = 1e3, d = 3, k = 2, means = mean_list, covs = cov)
fit <- FitGMM(data, k = 2, maxit = 10, eps = 1e-8)

cat("\n")
show(fit)

cat("Cluster means:\n")
fit@Means

cat("Cluster covariances:\n")
fit@Covariances

cat("Cluster responsibilities:\n")
head(fit@Responsibilities)

cat("\nCluster assignments:\n")
head(fit@Assignments)

cat("\nCompleted data:\n")
head(fit@Completed)
```

```
## Objective increment:  0.627
## Objective increment:  0.0673
## Objective increment:  0.0175
## Objective increment:  0.00462
## Objective increment:  0.00124
## Objective increment:  0.000331
## Objective increment:  8.86e-05
## Objective increment:  2.37e-05
## Objective increment:  6.37e-06
## Objective increment:  1.71e-06
## 10 update(s) performed without reaching tolerance limit.
##
##
## Gaussian Mixture Model with 2 Components.
##
## Cluster Proportions:
##    k1    k2
## 0.514 0.486
##
## Final Objective:
## [1] -3019.68
##
```

```
## Cluster means:
## [[1]]
##       y1       y2       y3
## 1.993715 2.044410 2.042874
##
## [[2]]
##        y1        y2        y3
## -2.023218 -1.967009 -2.031333
##
## Cluster covariances:
## [[1]]
##           y1        y2        y3
## y1 1.0421717 0.5260849 0.5547371
## y2 0.5260849 1.0938308 0.5580000
## y3 0.5547371 0.5580000 0.9795204
##
## [[2]]
##           y1        y2        y3
## y1 0.9486886 0.4428439 0.4698537
## y2 0.4428439 0.9356663 0.4986851
## y3 0.4698537 0.4986851 1.0774598
##
## Cluster responsibilities:
##             k1           k2
## 1 7.213209e-06 9.999928e-01
## 2 9.999999e-01 1.338099e-07
## 2 9.999770e-01 2.298827e-05
## 2 8.135063e-02 9.186494e-01
## 1 5.636022e-06 9.999944e-01
## 2 1.000000e+00 3.279117e-08
##
## Cluster assignments:
##   Assignments      Entropy
## 1           2 1.336147e-04
## 2           1 3.248369e-06
## 2           1 3.873851e-04
## 2           2 4.069204e-01
## 1           2 1.064057e-04
## 2           1 8.625655e-07
##
## Completed data:
##           y1         y2         y3
## 1 -3.5781991 -1.7243246 -0.7998434
## 2  2.4957664  3.1465942  1.6865892
## 2  1.4038862  1.7525323  2.0079278
## 2 -0.8862729 -1.2629163  1.0069399
## 1 -1.8648070 -0.7268469 -3.3138250
## 2  2.4270751  2.6503247  3.3227434
```

`mean`, `vcov`, and `logLik` methods are also defined for objects of class `mix`:

```
cat("\nObject class:\n")
class(fit)

cat("\nExtract mean:\n")
```

```r
mean(fit)

cat("\nExtract covariance:\n")
vcov(fit)

cat("\nExtract EM objective:\n")
logLik(fit)
```

```
##
## Object class:
## [1] "mix"
## attr(,"package")
## [1] "MGMM"
##
## Extract mean:
## [[1]]
##       y1       y2       y3
## 1.993715 2.044410 2.042874
##
## [[2]]
##        y1        y2        y3
## -2.023218 -1.967009 -2.031333
##
##
## Extract covariance:
## [[1]]
##           y1        y2        y3
## y1 1.0421717 0.5260849 0.5547371
## y2 0.5260849 1.0938308 0.5580000
## y3 0.5547371 0.5580000 0.9795204
##
## [[2]]
##           y1        y2        y3
## y1 0.9486886 0.4428439 0.4698537
## y2 0.4428439 0.9356663 0.4986851
## y3 0.4698537 0.4986851 1.0774598
##
##
## Extract EM objective:
## [1] -3019.684
```

Four Components with Missingness

In this example, 10^3 observations are simulated from a four-component bivariate normal distribution with 10% missingness.

```r
set.seed(200)

# Four components with missingness.
mean_list <- list(
  c(2, 2),
  c(2, -2),
  c(-2, 2),
```

```
  c(-2, -2)
)
sigma <- 0.5 * diag(2)
props <- c(0.35, 0.15, 0.15, 0.35)
data <- rGMM(
  n = 1000,
  d = 2,
  k = 4,
  pi = props,
  miss = 0.1,
  means = mean_list,
  covs = sigma
)
fit <- FitGMM(data, k = 4, maxit = 10, eps = 1e-8)
show(fit)

cat("Cluster means:\n")
fit@Means

cat("Cluster covariances:\n")
fit@Covariances

cat("\nCluster assignments:\n")
head(fit@Assignments)
```

```
## Objective increment:  1.57
## Objective increment:  0.161
## Objective increment:  0.0334
## Objective increment:  0.00726
## Objective increment:  0.0016
## Objective increment:  0.000341
## Objective increment:  7.46e-05
## Objective increment:  1.59e-05
## Objective increment:  3.48e-06
## Objective increment:  7.47e-07
## 10 update(s) performed without reaching tolerance limit.
##
## Gaussian Mixture Model with 4 Components.
##
## Cluster Proportions:
##    k1    k2    k3    k4
## 0.149 0.388 0.130 0.332
##
## Final Objective:
## [1] -1888.25
##
## Cluster means:
## [[1]]
##        y1        y2
## -2.046472  2.074567
##
## [[2]]
##        y1        y2
## -2.026168 -1.983955
```

```
##
## [[3]]
##        y1        y2
##  2.026073 -2.075973
##
## [[4]]
##       y1       y2
## 1.973530 1.981959
##
## Cluster covariances:
## [[1]]
##            y1         y2
## y1 0.43656741 0.02040745
## y2 0.02040745 0.45492340
##
## [[2]]
##             y1          y2
## y1 0.567726041 0.003079349
## y2 0.003079349 0.575620933
##
## [[3]]
##             y1          y2
## y1  0.39075316 -0.06937548
## y2 -0.06937548  0.37947376
##
## [[4]]
##             y1          y2
## y1 0.484058646 0.002193949
## y2 0.002193949 0.525919652
##
##
## Cluster assignments:
##   Assignments      Entropy
## 1           4 2.226854e-10
## 4           2 4.292356e-01
## 2           3 6.473445e-02
## 2           3 1.011832e-07
## 1           4 2.032369e-08
## 4           2 3.292494e-04
```

Cluster Number Selection

Clustering Quality

The function `ClustQual` provides several metrics for internally assessing the quality of cluster assignments from a fitted GMM. The input is an object of class `mix`. The output is a list containing the metrics: BIC, CHI, DBI, and SIL.

- BIC is the Bayesian Information Criterion, which is a penalized version of the negative log likelihood. A lower value indicates better clustering quality.
- CHI is the Calinski-Harabaz Index, a ratio of the between cluster to within cluster variation. A higher value indicates better clustering quality.
- DBI is the Davies-Bouldin Index, an average of cluster similarities. A lower value indicates better clustering quality.
- SIL is the average Silhouette width, a measure of how well an observation matches its assigned cluster. A higher value indicates better clustering quality.

```
set.seed(105)

# Four components without missingness
mean_list <- list(
  c(2, 2),
  c(2, -2),
  c(-2, 2),
  c(-2, -2)
)
cov <- 0.5 * diag(2)
data <- rGMM(n = 100, d = 2, k = 4, means = mean_list)
fit <- FitGMM(data, k = 4, maxit = 100, eps = 1e-8, report = FALSE)

# Quality metrics.
clust_qual <- ClustQual(fit)

cat("BIC:\n")
clust_qual$BIC

cat("\nCHI:\n")
clust_qual$CHI

cat("\nDBI:\n")
clust_qual$DBI

cat("\nSIL:\n")
clust_qual$SIL
```

```
## BIC:
## [1] 358.6248
##
## CHI:
## [1] 6.546848
##
## DBI:
## [1] 0.5617234
```

```
##
## SIL:
## [1] 0.545614
```

Choosing the Number of Clusters

In applications, the number of clusters k is often unknown. The function `ChooseK` is designed to assist in choosing the number of clusters. The inputs include the data matrix `data`, the minimum cluster number to assess `k0`, the maximum cluster number to assess `k1`, and the number of bootstrap replicates at each cluster number `boot`. For each cluster number k between k_0 and k_1, `boot` bootstrap data sets are generated. A GMM with k components is fit, and the quality metrics are calculated. The bootstrap replicates are summarized by their mean and standard error (SE). For each quality metric, the cluster number k_{opt} that had the optimal quality, and the smallest cluster number whose quality was within 1 SE of the optimum $k_{1\text{se}}$, are reported. The output is a list `Choices` containing the cluster numbers selected by each metric, and the complete set of bootstrap `Results`. Empirically, we find the silhouette width often performs well at identifying the number of clusters.

```r
# Cluster number selection.
choose_k <- ChooseK(data, k0 = 2, k1 = 6, boot = 10)

cat("\nCluster number choices:\n")
choose_k$Choices

cat("\nAll results:\n")
head(choose_k$Results)
```

```
## Cluster size 2 complete. 11 fit(s) succeeded.
## Cluster size 3 complete. 11 fit(s) succeeded.
## Cluster size 4 complete. 11 fit(s) succeeded.
## Cluster size 5 complete. 11 fit(s) succeeded.
## Cluster size 6 complete. 11 fit(s) succeeded.
##
## Cluster number choices:
##   Metric k_opt  Metric_opt k_1se  Metric_1se
## 1    BIC     6 289.8829238     5 323.7702176
## 2    CHI     6  10.2082679     6  10.2082679
## 3    DBI     4   0.5584643     4   0.5584643
## 4    SIL     4   0.5657591     4   0.5657591
##
## All results:
##   Clusters Fits Metric        Mean          SE
## 1        2   11    BIC 449.9947433 17.35825831
## 2        2   11    CHI   1.6024639  0.13045027
## 3        2   11    DBI   1.0222610  0.04206780
## 4        2   11    SIL   0.4372601  0.01385224
## 5        3   11    BIC 369.7079292 31.27016958
## 6        3   11    CHI   3.4225779  0.37144850
```

Multiple Imputation

`fit@Completed` contains a singly-imputed version of the input data, with missing values imputed to their posterior expectations. Although singly-imputed data are useful for some tasks, such as visualization, inference after single imputation is generally not valid. In *multiple imputation*, several completions of the input data are formed by drawing imputations for the missing values conditional on the observed values. The function `GenImputation` generates a single stochastic imputation of the input data from a fitted GMM, and may be called multiple times to generate multiple imputations of the data set.

Example

In the following example, `n = 100` observations are generated from a single component GMM (i.e. a multivariate normal distribution) with mean zero, identity covariance, and 10% of elements missing. A single component GMM is fit to the observed data. `m = 50` multiple imputations are performed. For each imputation, the marginal mean and its sampling variance (square of the standard error) are calculated and stored in respective lists. The function `CombineMIs` uses Rubin's rules to combine the lists of point estimates and sampling variances. The overall mean and sampling variance are presented. Finally, a χ^2_2 test is performed to assess the null hypothesis that the marginal mean is equal to zero, which is in fact true. Based on the p-value, the χ^2_2 test correctly fails to reject the null hypothesis.

```
# Generate data.
set.seed(103)
data <- rGMM(n = 100, d = 2, k = 1, miss = 0.1)

# Fit GMM.
fit <- FitGMM(data, k = 1, report = FALSE)

# Perform multiple imputation.
points <- list()
covs <- list()

m <- 50
for (i in seq_len(m)) {
  imputed <- GenImputation(fit)
  points[[i]] <- apply(imputed, 2, mean)
  covs[[i]] <- cov(imputed) / nrow(imputed)
}

# Combine point estimates and standard errors.
cat("Overall mean and its sampling variance:\n")
final <- CombineMIs(points, covs)
lapply(final, function(x) {round(x, digits = 3)})

# Is the overall mean significant different from zero?
x <- final$point
v <- final$cov
chi2_stat <- as.numeric(t(x) %*% solve(v, x))
pval <- pchisq(q = chi2_stat, df = 2, lower.tail = FALSE)

cat("\nP-value evaluating whether the mean is equal to zero:\n")
round(pval, digits = 3)
```

```
## Overall mean and its sampling variance:
## $point
```

```
##    y1    y2
## 0.059 0.043
##
## $cov
##       y1    y2
## y1 0.012 0.000
## y2 0.000 0.012
##
##
## P-value evaluating whether the mean is equal to zero:
## [1] 0.801
```