# Package ‘MoEClust’

May 12, 2020

**Type** Package

**Date** 2020-05-12

**Title** Gaussian Parsimonious Clustering Models with Covariates and a Noise Component

**Version** 1.3.1

**Description** Clustering via parsimonious Gaussian Mixtures of Experts using the MoEClust models introduced by Murphy and Murphy (2019) <doi:10.1007/s11634-019-00373-8>. This package fits finite Gaussian mixture models with a formula interface for supplying gating and/or expert network covariates using a range of parsimonious covariance parameterisations from the GPCM family via the EM/CEM algorithm. Visualisation of the results of such models using generalised pairs plots and the inclusion of an additional noise component is also facilitated. A greedy forward stepwise search algorithm is provided for identifying the optimal model in terms of the number of components, the GPCM covariance parameterisation, and the subsets of gating/expert network covariates.

**Depends** R (>= 4.0.0)

**License** GPL (>= 2)

**Encoding** UTF-8

**URL** https://cran.r-project.org/package=MoEClust

**BugReports** https://github.com/Keefe-Murphy/MoEClust

**LazyData** true

**Imports** lattice, matrixStats, mclust (>= 5.1), mvnfast, nnet, vcd

**Suggests** cluster, clustMD, geometry, knitr, rmarkdown, snow

**RoxygenNote** 7.1.0

**VignetteBuilder** knitr

**NeedsCompilation** no

**Author** Keefe Murphy [aut, cre] (<https://orcid.org/0000-0002-7709-3159>), Thomas Brendan Murphy [ctb] (<https://orcid.org/0000-0002-5668-7046>)

**Maintainer** Keefe Murphy <keefe.murphy@ucd.ie>

**Repository** CRAN

**Date/Publication** 2020-05-12 17:00:12 UTC

# R topics documented:

MoEClust-package . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 2
ais . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 5
aitken . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 6
as.Mclust . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 7
CO2data . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 9
drop_constants . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 10
drop_levels . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 11
expert_covar . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 12
force_posiDiag . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 14
MoE_clust . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 14
MoE_compare . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 21
MoE_control . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 25
MoE_crit . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 32
MoE_cstep . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 34
MoE_dens . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 36
MoE_estep . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 37
MoE_gpairs . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 39
MoE_mahala . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 45
MoE_news . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 47
MoE_plotCrit . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 48
MoE_plotGate . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 49
MoE_plotLogLik . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 50
MoE_stepwise . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 51
MoE_Uncertainty . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 54
noise_vol . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 56
plot.MoEClust . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 57
predict.MoEClust . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 59
quant_clust . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 62

**Index** **64**

---

`MoEClust-package` *MoEClust: Gaussian Parsimonious Clustering Models with Covariates and a Noise Component*

---

### Description

Clustering via parsimonious Gaussian Mixtures of Experts using the *MoEClust* models introduced by Murphy and Murphy (2019) <doi:10.1007/s11634-019-00373-8>. This package fits finite Gaussian mixture models with gating and/or expert network covariates using a range of parsimonious covariance parameterisations from the GPCM family via the EM/CEM algorithm. Visualisation of the results of such models using generalised pairs plots and the inclusion of an additional noise component is also facilitated.

## Usage

The most important function in the **MoEClust** package is: `MoE_clust`, for fitting the model via EM/CEM with gating and/or expert network covariates, supplied via formula interfaces.

`MoE_compare` is provided for conducting model selection between different results from `MoE_clust` using different covariate combinations &/or initialisation strategies, etc.

`MoE_stepwise` is provided for conducting a greedy forward stepwise search to identify the optimal model in terms of the number of components, GPCM covariance type, and the subsets of gating/expert network covariates.

`MoE_control` allows supplying additional arguments to `MoE_clust` and `MoE_stepwise` which govern, among other things, controls on the inclusion of an additional noise component and controls on the initialisation of the allocations for the EM/CEM algorithm.

A dedicated plotting function (`plot.MoEClust`) exists for visualising the results using generalised pairs plots, for examining the gating network, and/or log-likelihood, and/or clustering uncertainties, and/or graphing model selection criteria values. The generalised pairs plots (`MoE_gpairs`) visualise all pairwise relationships between clustered response variables and associated continuous, categorical, and/or ordinal covariates in the gating &/or expert networks, coloured according to the MAP classification, and also give the marginal distributions of each variable (incl. the covariates) along the diagonal.

An `as.Mclust` method is provided to coerce the output of class `"MoEClust"` from `MoE_clust` to the `"Mclust"` class, to facilitate use of plotting and other functions for the `"Mclust"` class within the **mclust** package. As per **mclust**, **MoEClust** also facilitates modelling with an additional noise component (with or without the mixing proportion for the noise component depending on covariates).

Finally, a `predict` method is provided for predicting the fitted response and probability of cluster membership (and by extension the MAP classification) for new data, in the form of new covariates and new response data, or new covariates only.

Other functions also exist, e.g. `MoE_crit`, `MoE_dens`, `MoE_estep`, `MoE_compare`, and `aitken`, which are all used within `MoE_clust` but are nonetheless made available for standalone use.

The package also contains two data sets: `ais` and `CO2data`.

## Details

- Type: Package
- Package: MoEClust
- Version: 1.3.1
- Date: 2020-05-12 (this version), 2017-11-28 (original release)
- Licence: GPL (>=2)

## See Also

Further details and examples are given in the associated vignette document:
`vignette("MoEClust",package = "MoEClust")`

### Author(s)

Keefe Murphy [aut, cre], Thomas Brendan Murphy [ctb]

**Maintainer**: Keefe Murphy - <<keefe.murphy@ucd.ie>>

### References

Murphy, K. and Murphy, T. B. (2019). Gaussian parsimonious clustering models with covariates and a noise component. *Advances in Data Analysis and Classification*, 1-33. <doi:10.1007/s11634-019-00373-8>.

### See Also

Useful links:

- https://cran.r-project.org/package=MoEClust
- Report bugs at https://github.com/Keefe-Murphy/MoEClust

### Examples

```
data(ais)

# Fit two models
res1  <- MoE_clust(ais[,3:7], G=2, gating=~BMI, expert=~sex,
                   modelNames=c("EVE", "VVE", "VEE"), network.data=ais)
res2  <- MoE_clust(ais[,3:7], G=2, equalPro=TRUE, expert=~sex,
                   modelNames=c("EVE", "VVE", "VEE"), network.data=ais)

# Compare the best model from each set of results
(comp <- MoE_compare(res1, res2, optimal.only=TRUE))

# Produce a plot for the optimal model
plot(comp$optimal, what="gpairs")

# Summarise its classification table, component parameters, and gating/expert networks
summary(comp$optimal, classification=TRUE, parameters=TRUE, networks=TRUE)

data(CO2data)
CO2   <- CO2data$CO2
GNP   <- CO2data$GNP

# Fit a range of models
m1    <- MoE_clust(CO2, G=1:3)
m2    <- MoE_clust(CO2, G=2:3, gating= ~ GNP)
m3    <- MoE_clust(CO2, G=1:3, expert= ~ GNP)
m4    <- MoE_clust(CO2, G=2:3, gating= ~ GNP, expert= ~ GNP)
m5    <- MoE_clust(CO2, G=2:3, equalPro=TRUE)
m6    <- MoE_clust(CO2, G=2:3, expert= ~ GNP, equalPro=TRUE)

# Extract the model with highest BIC
(comp <- MoE_compare(m1, m2, m3, m4, m5, m6, criterion="bic"))
```

```
# See if a better model can be found using greedy forward stepwise selection
# Conduct a stepwise search on the same data
(mod1 <- MoE_stepwise(CO2, CO2data[,"GNP", drop=FALSE]))

# Conduct another stepwise search considering models with a noise component
(mod2 <- MoE_stepwise(CO2, CO2data[,"GNP", drop=FALSE], noise=TRUE))

# Compare all sets of results to choose the optimal model
(best <- MoE_compare(mod1, mod2, comp, pick=1)$optimal)
```

---

`ais` *Australian Institute of Sport data*

---

## Description

Data on 102 male and 100 female athletes collected at the Australian Institute of Sport, courtesy of Richard Telford and Ross Cunningham.

## Usage

```
data(ais)
```

## Format

A data frame with 202 observations on the following 13 variables:

- [,1] `sex` - categorical, levels = `female,male`
- [,2] `sport` - categorical, levels = `B_Ball,Field,Gym,Netball,Row,Swim,T_400m,Tennis,T_Sprnt,W_Polo`
- [,3] `RCC` - red cell count (numeric)
- [,4] `WCC` - white cell count (numeric)
- [,5] `Hc` - Hematocrit (numeric)
- [,6] `Hg` - Hemoglobin (numeric)
- [,7] `Fe` - plasma ferritin concentration (numeric)
- [,8] `BMI` - body mass index: `Wt/(Ht)^2` (numeric)
- [,9] `SSF` - sum of skin folds (numeric)
- [,10] `Bfat` - body fat percentage (numeric)
- [,11] `LBM` - lean body mass (numeric)
- [,12] `Ht` - height, cm (numeric)
- [,13] `Wt` - weight, kg (numeric)

## Details

The data have been made publicly available in connection with the book by Cook and Weisberg (1994).

### References

Cook, R. D. and Weisberg, S. (1994), *An Introduction to Regression Graphics*. John Wiley & Sons, New York.

### Examples

```
data(ais, package="MoEClust")
pairs(ais[,c(3:7)], col=as.numeric(ais$sex), main = "AIS data")
apply(ais[,c(3:7)], 2, summary)
```

---

## aitken *Aitken Acceleration*

---

### Description

Calculates the Aitken acceleration estimate of the final converged maximised log-likelihood under the EM/CEM framework.

### Usage

```
aitken(loglik)
```

### Arguments

| | |
|---|---|
| `loglik` | A vector of three consecutive log-likelihood values. These three values should be in ascending order, though this is not checked. |

### Details

The final converged maximised log-likelihood can be used to determine convergence of the EM/CEM algorithm within `MoE_clust`, i.e. by checking whether the absolute difference between the current log-likelihood estimate and the final converged maximised log-likelihood estimate is less than some tolerance.

### Value

A list with the following named components:

| | |
|---|---|
| `ll` | The most current estimate for the log-likelihood. |
| `linf` | The most current estimate of the final converged maximised log-likelihood. |
| `a` | The Aitken acceleration value where typically `0 <= a <= 1`. When `a < 0`, a numerical issue or bug has occurred; when `a > 1`, the algorithm is accelerating and should not be stopped. |

When the `"aitken"` method is employed within `MoE_clust` (via `MoE_control`), `ll` at convergence gives the log-likelihood achieved by the estimated parameters, while `linf` at convergence estimates the log-likelihood that would be achieved after an infinite number of EM/CEM iterations.

## Note

Within MoE_clust, as specified by the stopping argument of MoE_control, "aitken" is the default method used to assess convergence. The other option monitors the "relative" change in log-likelihood against some tolerance. See MoE_control.

## Author(s)

Keefe Murphy - <<keefe.murphy@ucd.ie>>

## References

Boehning, D., Dietz, E., Schaub, R., Schlattmann, P. and Lindsay, B. G. (1994). The distribution of the likelihood ratio for mixtures of densities from the one-parameter exponential family. *Annals of the Institute of Statistical Mathematics*, 46(2): 373-388.

## See Also

MoE_control

## Examples

```
(a1 <- aitken(-c(449.61534, 442.84221, 436.58999)))
a2  <- aitken(-c(442.84221, 436.58999, 436.58998))
abs(a2$linf - a1$linf) < 1e-05 #FALSE
a3  <- aitken(-c(436.58998, 436.58997, 436.58997))
abs(a3$linf - a2$linf) < 1e-05 #TRUE
(ll <- a3$linf)
(a  <- a3$a)
```

---

# as.Mclust *Convert MoEClust objects to the Mclust class*

---

## Description

Converts an object of class "MoEClust" generated by MoE_clust and converts it to an object of class "Mclust" as generated by fitting Mclust, to facilitate use of plotting and other functions for the "Mclust" class within the **mclust** package. Some caution is advised when converting models with gating &/or expert covariates (see Note below).

## Usage

```
## S3 method for class 'MoEClust'
as.Mclust(x,
          expert.covar = TRUE,
          signif = 0L,
          ...)
```

### Arguments

| | |
|---|---|
| x | An object of class "MoEClust" generated by MoE_clust or an object of class "MoECompare" generated by MoE_compare. Models with a noise component are facilitated here too. |
| expert.covar | Logical (defaults to TRUE) governing whether the extra variability in the component means is added to the MVN ellipses corresponding to the component covariance matrices in the presence of expert network covariates. See the function expert_covar. |
| signif | Significance level for outlier removal. Must be a single number in the interval [0, 1). Corresponds to the percentage of data to be considered extreme and therefore removed (half of signif at each endpoint, on a column-wise basis). The default, 0, corresponds to no outlier removal. **Only** invoke this argument as an aid to visualisation via plot.Mclust. |
| ... | Further arguments to be passed to other methods. |

### Details

Of course, the user is always encouraged to use the dedicated plot function for objects of the "MoEClust" class instead, but calling plot after converting via as.Mclust can be particularly useful for univariate mixtures.

In the presence of expert network covariates, the component-specific covariance matrices are (by default, via the argument expert.covar) modified for plotting purposes via the function expert_covar, in order to account for the extra variability of the means, usually resulting in bigger shapes & sizes for the MVN ellipses.

The signif argument is intended only to aid visualisation via plot.Mclust, as plots therein can be sensitive to outliers, particularly with regard to axis limits.

### Value

An object of class "Mclust". See methods(class="Mclust") for a (non-exhaustive) list of functions which can be applied to this class.

### Note

Of the functions which can be applied to the result of the conversion, logLik.Mclust shouldn't be trusted in the presence of either expert network covariates, or (for models with more than 1 component) gating network covariates.

Mixing proportions are averaged over observations in components in the presence of gating network covariates during the coercion.

Plots may be quite misleading in the presence of gating &/or expert covariates when the what argument is "density" within plot.Mclust; users are **strongly** encouraged to use MoE_gpairs with response.type="density" instead.

The functions clustCombi and clustCombiOptim can be safely used (provided as.Mclust(x) is supplied as the object argument to clustCombi), as they only rely on x$z and x$G only. See the examples below.

### Author(s)

Keefe Murphy - <<keefe.murphy@ucd.ie>>

### References

Fraley, C. and Raftery, A. E. (2002). Model-based clustering, discriminant analysis, and density estimation. *Journal of the American Statistical Association*, 97(458):611-631.

Scrucca L., Fop M., Murphy T. B. and Raftery A. E. (2016). mclust 5: clustering, classification and density estimation using Gaussian finite mixture models. *The R Journal*, 8(1):289-317.

### See Also

Mclust, plot.Mclust, MoE_clust, plot.MoEClust, expert_covar

### Examples

```
# library(mclust)

# Fit a gating network mixture of experts model to the ais data
# data(ais)
# mod   <- MoE_clust(ais[,3:7], G=1:9, gating= ~ BMI + sex, network.data=ais)

# Convert to the "Mclust" class and examine the classification
# mod2  <- as.Mclust(mod)
# plot(mod2, what="classification")

# Examine the uncertainty
# plot(mod2, what="uncertainty")

# Return the optimal number of clusters according to entropy
# combi <- mclust::clustCombi(object=mod2)
# optim <- mclust::clustCombiOptim(combi)
# table(mod2$classification, ais$sex)
# table(optim$cluster.combi, ais$sex)

# While we could have just used plot.MoEClust above,
# plot.Mclust is especially useful for univariate data
# data(CO2data)
# res <- MoE_clust(CO2data$CO2, G=3, equalPro=TRUE, expert = ~ GNP, network.data=CO2data)
# plot(as.Mclust(res))
```

---

## CO2data — *GNP and CO2 Data Set*

---

### Description

This data set gives the gross national product (GNP) per capita in 1996 for various countries as well as their estimated carbon dioxide ($CO_2$) emission per capita for the same year.

## Usage

```
data(CO2data)
```

## Format

This data frame consists of 28 countries and the following variables:

- `GNP` - The gross product per capita in 1996.
- `CO2` - The estimated carbon dioxide emission per capita in 1996.
- `country` - An abbreviation pertaining to the country measures (e.g. `"GRC"` = Greece and `"CH"` = Switzerland).

## References

Hurn, M., Justel, A. and Robert, C. P. (2003) Estimating mixtures of regressions, *Journal of Computational and Graphical Statistics*, 12(1): 55-79.

## Examples

```
data(CO2data, package="MoEClust")
plot(CO2data$GNP, CO2data$CO2, type="n", ylab=expression('CO'[2]))
text(CO2data$GNP, CO2data$CO2, CO2data$country)
```

---

# `drop_constants` *Drop constant variables from a formula*

---

## Description

Drops constant variables from the RHS of a formula taking the data set (`dat`), the formula (`formula`), and an optional subset vector (`sub`) as arguments.

## Usage

```
drop_constants(dat,
               formula,
               sub = NULL)
```

## Arguments

| | |
|---|---|
| `dat` | A `data.frame` where rows correspond to observations and columns correspond to variables. Ideally column names should be present. |
| `formula` | An object of class `"formula"`: a symbolic description of the model to be fitted. Variables in the `formula` not present in the columns of `dat` will automatically be discarded. The `formula` may include interactions, transformations, or higher order terms: the latter **must** be specified explicitly using the `AsIs` operator (`I`). |
| `sub` | An optional vector specifying a subset of observations to be used in the fitting process. |

### Value

The updated formula with constant variables removed.

### Note

Formulas with and without intercepts are accommodated.

### Author(s)

Keefe Murphy - <<keefe.murphy@ucd.ie>>

### See Also

drop_levels, I

### Examples

```
data(ais)
hema  <- as.matrix(ais[,3:7])
sex   <- ais$sex
BMI   <- ais$BMI

# Set up a no-intercept regression formula with constant column 'sex'
form1 <- as.formula(hema ~ sex + BMI + I(BMI^2) - 1)
sub   <- ais$sex == "male"

# Try fitting a linear model
mod1  <- try(lm(form1, data=ais, subset=sub), silent=TRUE)
mod1  <- try(lm(form1, data=ais, subset=sub), silent=TRUE)
inherits(mod1, "try-error") # TRUE

# Remove redundant variables from formula & try again
form2 <- drop_constants(ais, form1, sub)
mod2  <- try(lm(form2, data=ais, subset=sub), silent=TRUE)
inherits(mod2, "try-error") # FALSE
```

---

## drop_levels — *Drop unused factor levels to predict from unseen data*

---

### Description

Drops unseen factor levels in newdata for which predictions are required from a lm model fit.

### Usage

```
drop_levels(fit,
            newdata)
```

### Arguments

| | |
|---|---|
| fit | A fitted lm model. |
| newdata | A data.frame containing variables with which to predict. |

### Value

A data.frame like newdata with unseen factor levels replaced by NA.

### Note

This function is untested for models other than lm.

### Author(s)

Keefe Murphy - <<keefe.murphy@ucd.ie>>

### See Also

drop_constants

### Examples

```
data(ais)
hema  <- as.matrix(ais[,3:7])
BMI   <- ais$BMI
sport <- ais$sport
sub   <- ais$sport != "Row"

# Fit a linear model
mod   <- lm(hema ~ BMI + sport, data=ais, subset=sub)

# Make predictions
pred1 <- try(predict(mod, newdata=ais), silent=TRUE)
inherits(pred1, "try-error") #TRUE

# Remove unused levels and try again
pred2 <- try(predict(mod, newdata=drop_levels(mod, ais)), silent=TRUE)
inherits(pred2, "try-error") #FALSE
anyNA(pred2)                 #TRUE
```

---

## expert_covar — *Account for extra variability in covariance matrices with expert covariates*

---

### Description

In the presence of expert network covariates, this helper function modifies the component-specific covariance matrices of a "MoEClust" object, in order to account for the extra variability of the means, usually resulting in bigger shapes & sizes for the MVN ellipses. The function also works for univariate response data.

### Usage

```
expert_covar(x)
```

### Arguments

x — An object of class "MoEClust" generated by MoE_clust, or an object of class "MoECompare" generated by MoE_compare. Models with a noise component are facilitated here too.

### Details

This function is used internally by plot.MoEClust and as.Mclust, for visualisation purposes.

### Value

The variance component only from the parameters list from the output of a call to MoE_clust, modified accordingly.

### Note

The modelName of the resulting variance object may not correspond to the model name of the "MoEClust" object, in particular scale, shape, &/or orientation may no longer be constrained across clusters. Usually, the modelName of the transformed variance object will be "VVV".

### Author(s)

Keefe Murphy - <<keefe.murphy@ucd.ie>>

### References

Murphy, K. and Murphy, T. B. (2019). Gaussian parsimonious clustering models with covariates and a noise component. *Advances in Data Analysis and Classification*, 1-33. <doi:10.1007/s11634-019-00373-8>.

### See Also

MoE_clust, MoE_gpairs, plot.MoEClust, as.Mclust

### Examples

```
data(ais)
res   <- MoE_clust(ais[,3:7], G=2, gating= ~ 1, expert= ~ sex,
                   network.data=ais, modelNames="EEE", equalPro=TRUE)

# Extract the variance object
res$parameters$variance

# Modify the variance object
expert_covar(res)
```

## force_posiDiag *Force diagonal elements of a triangular matrix to be positive*

### Description

This function ensures that the triangular matrix in a QR (or other) decomposition has positive values along its diagonal.

### Usage

```
force_posiDiag(x)
```

### Arguments

| | |
|---|---|
| x | A matrix, which must be either upper-triangular or lower-triangular. |

### Value

An upper or lower triangular matrix with positive diagonal entries such that the matrix is still a valid decomposition of the matrix the input x is a decomposition of.

### Author(s)

Keefe Murphy - <<keefe.murphy@ucd.ie>>

### Examples

```
data(ais)
res <- MoE_clust(ais[,3:7], G=3, modelNames="EEE")
sig <- res$parameters$variance
a   <- force_posiDiag(sig$cholSigma)
b   <- chol(sig$Sigma)
round(sum(a - b), 10) == 0          #TRUE
sum(crossprod(a) != sig$Sigma) == 0 #TRUE
sum(crossprod(b) != sig$Sigma) == 0 #TRUE
```

## MoE_clust *MoEClust: Gaussian Parsimonious Clustering Models with Covariates and a Noise Component*

### Description

Fits MoEClust models: Gaussian Mixture of Experts models with GPCM/**mclust**-family covariance structures. In other words, performs model-based clustering via the EM/CEM algorithm where covariates are allowed to enter neither, either, or both the mixing proportions (gating network) and/or component densities (expert network) of a Gaussian Parsimonious Clustering Model, with or without an additional noise component. Additional arguments are available via the function MoE_control, including the specification of a noise component, controls on the initialisation of the algorithm, and more.

### Usage

```
MoE_clust(data,
          G = 1:9,
          modelNames = NULL,
          gating = ~1,
          expert = ~1,
          control = MoE_control(...),
          network.data = NULL,
          ...)

## S3 method for class 'MoEClust'
print(x,
      digits = 3L,
      ...)

## S3 method for class 'MoEClust'
summary(object,
        classification = TRUE,
        parameters = FALSE,
        networks = FALSE,
        ...)
```

### Arguments

`data` A numeric vector, matrix, or data frame of observations. Categorical variables are not allowed. If a matrix or data frame, rows correspond to observations and columns correspond to variables.

`G` An integer vector specifying the numbers of mixture components (clusters) to fit. Defaults to `G=1:9`. Must be a strictly positive integer, unless a noise component is included in the estimation, in which case `G=0` is allowed and included by default. (see `MoE_control`).

`modelNames` A vector of character strings indicating the models to be fitted in the EM/CEM phase of clustering. With `n` observations and `d` variables, the defaults are:

| | |
|---|---|
| for univariate data | `c("E", "V")` |
| for multivariate data $n > d$ | `mclust.options("emModelNames")` |
| for high-dimensional multivariate data $n \leq d$ | `c("EII", "VII", "EEI", "EVI", "VEI", "VVI")` |

For single-component models these options reduce to:

| | |
|---|---|
| for univariate data | `"E"` |
| for multivariate data $n > d$ | `c("EII", "EEI", "EEE")` |
| for high-dimensional multivariate data $n \leq d$ | `c("EII", "EEI")` |

For zero-component models with a noise component only the `"E"` and `"EII"` models will be fit for univariate and multivariate data, respectively. The help file for `mclustModelNames` further describes the available models (though the `"X"` in the single-component models will be coerced to "E" if supplied that way). For single-component models, other model names equivalent to those above can be supplied, but will be coerced to those above.

gating
: A formula for determining the model matrix for the multinomial logistic regression in the gating network when fixed covariates enter the mixing proportions. Defaults to ~1, i.e. no covariates. This will be ignored where G=1. Continuous, categorical, and/or ordinal covariates are allowed. Logical covariates will be coerced to factors. Interactions, transformations, and higher order terms are permitted: the latter **must** be specified explicitly using the AsIs operator (I). The specification of the LHS of the formula is ignored. Intercept terms are included by default.

expert
: A formula for determining the model matrix for the (multivariate) WLS in the expert network when fixed covariates are included in the component densities. Defaults to ~1, i.e. no covariates. Continuous, categorical, and/or ordinal covariates are allowed. Logical covariates will be coerced to factors. Interactions, transformations, and higher order terms are permitted: the latter **must** be specified explicitly using the AsIs operator (I). The specification of the LHS of the formula is ignored. Intercept terms are included by default.

control
: A list of control parameters for the EM/CEM and other aspects of the algorithm. The defaults are set by a call to MoE_control. In particular, arguments pertaining to the inclusion of an additional noise component are documented here.

network.data
: An optional data frame (or a matrix with named columns) in which to look for the covariates in the gating &/or expert network formulas, if any. If not found in network.data, any supplied gating &/or expert covariates are taken from the environment from which MoE_clust is called. Try to ensure the names of variables in network.data do not match any of those in data.

...
: An alternative means of passing control parameters directly via the named arguments of MoE_control. Do not pass the output from a call to MoE_control here! This argument is only relevant for the MoE_clust function and will be ignored for the associated print and summary functions.

x, object, digits, classification, parameters, networks
: Arguments required for the print and summary functions: x and object are objects of class "MoEClust" resulting from a call to MoE_clust, while digits gives the number of decimal places to round to for printing purposes (defaults to 3). classification, parameters, and networks are logicals which govern whether a table of the MAP classification of observations, the mixture component parameters, and the gating/expert network coefficients are printed, respectively.

## Details

The function effectively allows 6 different types of Gaussian Mixture of Experts model (as well as the different models in the GPCM/**mclust** family, for each): i) the standard finite Gaussian mixture with no covariates, ii) fixed covariates only in the gating network, iii) fixed covariates only in the expert network, iv) the full Mixture of Experts model with fixed covariates entering both the mixing proportions and component densities. By constraining the mixing proportions to be equal (see equalPro in MoE_control) two extra special cases are facilitated when gating covariates are excluded.

Note that having the same covariates in both networks is allowed. So too are interactions, transformations, and higher order terms (see formula): the latter **must** be specified explicitly using the AsIs operator (I). Covariates can be continuous, categorical, logical, or ordinal, but the response must always be continuous.

While model selection in terms of choosing the optimal number of components and the GPCM/**mclust** model type is performed within MoE_clust, using one of the criterion options within MoE_control, choosing between multiple fits with different combinations of covariates or different initialisation settings can be done by supplying objects of class "MoEClust" to MoE_compare.

## Value

A list (of class "MoEClust") with the following named entries, mostly corresponding to the chosen optimal model (as determined by the criterion within MoE_control):

- call: The matched call.
- data: The input data, as a data.frame.
- modelName: A character string denoting the GPCM/**mclust** model type at which the optimal criterion occurs.
- n: The number of observations in the data.
- d: The dimension of the data.
- G: The optimal number of mixture components according to criterion.
- BIC: A matrix of *all* BIC values with length{G} rows and length(modelNames) columns. May include missing entries: NA represents models which were not visited, -Inf represents models which were terminated due to error, for which a log-likelihood could not be estimated. Inherits the classes "MoECriterion" and "mclustBIC", for which dedicated printing and plotting functions exist, respectively.
- ICL: A matrix of *all* ICL values with length{G} rows and length(modelNames) columns. May include missing entries: NA represents models which were not visited, -Inf represents models which were terminated due to error, for which a log-likelihood could not be estimated. Inherits the classes "MoECriterion" and "mclustICL", for which dedicated printing and plotting functions exist, respectively.
- AIC: A matrix of *all* AIC values with length{G} rows and length(modelNames) columns. May include missing entries: NA represents models which were not visited, -Inf represents models which were terminated due to error, for which a log-likelihood could not be estimated. Inherits the classes "MoECriterion" and "mclustAIC", for which dedicated printing and plotting functions exist, respectively.
- bic: The BIC value corresponding to the optimal model. May not necessarily be the optimal BIC.
- icl: The ICL value corresponding to the optimal model. May not necessarily be the optimal ICL.

aic
: The AIC value corresponding to the optimal model. May not necessarily be the optimal AIC.

gating
: An object of class "MoE_gating" and either "multinom" or "glm" (for single-component models) giving the multinom regression coefficients of the gating network. If gating covariates were *NOT* supplied (or the best model has just one component), this corresponds to a RHS of ~1, otherwise the supplied gating formula. As such, a fitted gating network is always returned even in the absence of supplied covariates. The number of parameters to penalise by for MoE_crit is given by length(coef(gating)), and the gating formula used is stored here as an attribute. If there is a noise component (and the option noise.gate=TRUE is invoked), its coefficients are those for the *last* component. **Users are cautioned against making inferences about statistical significance from summaries of the coefficients in the gating network**.

expert
: An object of class "MoE_expert" and "lm" giving the (multivariate) WLS regression coefficients of the expert network. If expert covariates were NOT supplied, this corresponds to a RHS of ~1, otherwise the supplied expert formula. As such, a fitted expert network is always returned even in the absence of supplied covariates. The number of parameters to penalise by for MoE_crit is given by G * length(coef(expert[[1]])), and the expert formula used is stored here is an attribute. **Users are cautioned against making inferences about statistical significance from summaries of the coefficients in the expert network**.

LOGLIK
: A matrix of *all* maximal log-likelihood values with length{G} rows and length(modelNames) columns. May include missing entries: NA represents models which were not visited, -Inf represents models which were terminated due to error, for which a log-likelihood could not be estimated. Inherits the classes "MoECriterion" and "mclustLoglik", for which dedicated printing and plotting functions exist, respectively.

loglik
: The vector of increasing log-likelihood values for every EM/CEM iteration under the optimal model. The last element of this vector is the maximum log-likelihood achieved by the parameters returned at convergence.

linf
: An asymptotic estimate of the final converged maximised log-likelihood. Returned when stopping="aitken" and G > 1 (see MoE_control and aitken), otherwise the last element of loglik is returned instead.

df
: The number of estimated parameters in the optimal model (i.e. the number of 'used' degrees of freedom). Subtract this number from n to get the degrees of freedom. The number of parameters due to the gating network, expert network, and covariance matrices are also stored here as attributes of df.

iters
: The total number of EM/CEM iterations for the optimal model.

hypvol
: The hypervolume parameter for the noise component if required, otherwise set to NA (see MoE_control).

parameters
: A list with the following named components:

    pro
    : The mixing proportions: either a vector of length G or, if gating covariates were supplied, a matrix with an entry for each observation (rows) and component (columns).

`mean` The means of each component. If there is more than one component, this is a matrix whose $k$-th column is the mean of the $k$-th component of the mixture model.

For models with expert network covariates, this is given by the posterior mean of the fitted values, otherwise the posterior mean of the response is reported. For models with expert network covariates, the *observation-specific* means can be accessed by calling `predict` on each object in the list given by `expert`.

`variance` A list of variance parameters of each component of the model. The components of this list depend on the model type specification. See the help file for `mclustVariance` for details. Also see `expert_covar` for an alternative approach to summarising the variance parameters in the presence of expert network covariates.

`Vinv` The inverse of the hypervolume parameter for the noise component if required, otherwise set to `NULL` (see `MoE_control`).

`z` — The final responsibility matrix whose `[i,k]`-th entry is the probability that observation $i$ belonds to the $k$-th component. If there is a noise component, its values are found in the *last* column.

`classification` — The vector of cluster labels for the chosen model corresponding to `z`, i.e. `max.col(z)`. Observations belonging to the noise component, if any, will belong to component `0`.

`uncertainty` — The uncertainty associated with the `classification`.

`net.covs` — A data frame gathering the unique set of covariates used in the `gating` and `expert` networks, if any. Will contain zero columns in the absence of gating or expert network covariates. Supplied gating covariates will be excluded if the optimal model has only one component. May have fewer columns than covariates supplied via the `network.data` argument also, as only the included covariates are gathered here.

`resid.data` — In the presence of expert network covariates, this is the augmented data actually used in the clustering at convergence, as a list of `G` matrices of WLS residuals of dimension `n * d`. Will contain zero columns in the absence of expert network covariates.

`DF` — A matrix giving the numbers of estimated parameters (i.e. the number of 'used' degrees of freedom) for *all* visited models, with `length{G}` rows and `length(modelNames)` columns. Subtract these numbers from `n` to get the degrees of freedom. May include missing entries: `NA` represents models which were not visited, `-Inf` represents models which were terminated due to error, for which parameters could not be estimated. Inherits the classes `"MoECriterion"` and `"mclustDF"`, for which dedicated printing and plotting functions exist, respectively.

`ITERS` — A matrix giving the total number of EM/CEM iterations for *all* visited models, with `length{G}` rows and `length(modelNames)` columns. May include missing entries: `NA` represents models which were not visited, `Inf` represents models which were terminated due to singularity/error and thus would never have converged. Inherits the classes `"MoECriterion"` and `"mclustITERS"`, for which dedicated printing and plotting functions exist, respectively.

Dedicated plot, predict, print and summary functions exist for objects of class "MoEClust". The results can be coerced to the "Mclust" class to access other functions from the **mclust** package via as.Mclust.

## Note

Where BIC, ICL, AIC, LOGLIK, DF and ITERS contain NA entries, this corresponds to a model which was not run; for instance a VVV model is never run for single-component models as it is equivalent to EEE. As such, one can consider the value as not really missing, but equivalent to the EEE value. BIC, ICL, AIC, LOGLIK, DF and ITERS all inherit the classes "MoECriterion" and "mclustBIC","mclustICL",etc., for which dedicated printing and plotting functions exist, respectively.

## Author(s)

Keefe Murphy - <<keefe.murphy@ucd.ie>>

## References

Murphy, K. and Murphy, T. B. (2019). Gaussian parsimonious clustering models with covariates and a noise component. *Advances in Data Analysis and Classification*, 1-33. <doi:10.1007/s11634-019-00373-8>.

Fraley, C. and Raftery, A. E. (2002). Model-based clustering, discriminant analysis, and density estimation. *Journal of the American Statistical Association*, 97(458):611-631.

## See Also

See MoE_stepwise for identifying the optimal model and its covariates via greedy forward stepwise selection.

MoE_compare, plot.MoEClust, predict.MoEClust, MoE_control, as.Mclust, MoE_crit, MoE_estep, MoE_cstep, MoE_dens, mclustModelNames, mclustVariance, expert_covar, aitken, I

## Examples

```
data(ais)
hema  <- ais[,3:7]
sex   <- ais$sex
BMI   <- ais$BMI

# Fit a standard finite mixture model
m1    <- MoE_clust(hema, G=2:3)

# Allow covariates to enter the mixing proportions
m2    <- MoE_clust(hema, G=2:3, gating= ~ sex + BMI)

# Allow covariates to enter the component densities
m3    <- MoE_clust(hema, G=2:3, expert= ~ sex)

# Allow covariates to enter both the gating & expert network
```

```
m4    <- MoE_clust(hema, G=2:3, gating= ~ BMI, expert= ~ sex)

# Fit an equal mixing proportion model with an expert network covariate
m5    <- MoE_clust(hema, G=2:3, expert= ~ sex + BMI, equalPro=TRUE)

# Fit models with gating covariates & an additional noise component
m6    <- MoE_clust(hema, G=2:3, tau0=0.1, gating=~BMI, network.data=ais)

# Extract the model with highest BIC
(comp <- MoE_compare(m1, m2, m3, m4, m5, m6, criterion="bic"))

# See if a better model can be found using greedy forward stepwise selection
(step <- MoE_stepwise(ais[,3:7], ais))
(comp <- MoE_compare(comp, step, optimal.only=TRUE))
(best <- comp$optimal)
(summ <- summary(best, classification=TRUE, parameters=TRUE, networks=TRUE))

# Examine the expert network in greater detail
# (but refrain from inferring statistical significance!)
summary(best$expert)

# Visualise the results, incl. the gating network and log-likelihood
plot(best, what="gpairs")
plot(best, what="gating") # equal mixing proportions!
plot(best, what="loglik")

# Visualise the results using the 'lattice' library
require("lattice")
z     <- factor(best$classification, labels=paste0("Cluster", seq_len(best$G)))
splom(~ hema | sex, groups=z)
splom(~ hema | z, groups=sex)
```

---

`MoE_compare` *Choose the best MoEClust model*

---

## Description

Takes one or more sets of MoEClust models fitted by `MoE_clust` (or `MoE_stepwise`) and ranks them according to the BIC, ICL, or AIC. It's possible to respect the internal ranking within each set of models, or to discard models within each set which were already deemed sub-optimal. This function can help with model selection via exhaustive or stepwise searches.

## Usage

```
MoE_compare(...,
            criterion = c("bic", "icl", "aic"),
            pick = 10L,
            optimal.only = FALSE)

## S3 method for class 'MoECompare'
```

```
print(x,
      index = seq_len(x$pick),
      digits = 3L,
      details = TRUE,
      ...)
```

## Arguments

`...` One or more objects of class "MoEClust" outputted by MoE_clust. All models must have been fit to the same data set. A single *named* list of such objects can also be supplied. Additionally, objects of class "MoECompare" outputted by this very function can also be supplied here.

This argument is only relevant for the MoE_compare function and will be ignored for the associated `print` function.

`criterion` The criterion used to determine the ranking. Defaults to "bic".

`pick` The (integer) number of models to be ranked and compared. Defaults to `10L`. Will be constrained by the number of models within the "MoEClust" objects supplied via `...` if `optimal.only` is `FALSE`, otherwise constrained simply by the number of "MoEClust" objects supplied. Setting `pick=Inf` is a valid way to select all models.

`optimal.only` Logical indicating whether to only rank models already deemed optimal within each "MoEClust" object (`TRUE`), or to allow models which were deemed suboptimal enter the final ranking (`FALSE`, the default). See `details`.

`x, index, digits, details`
Arguments required for the associated `print` function:

- `x` An object of class "MoECompare" resulting from a call to MoE_compare.
- `index` A logical or numeric vector giving the indices of the rows of the table of ranked models to print. This defaults to the full set of ranked models. It can be useful when the table of ranked models is large to examine a subset via this `index` argument, for display purposes.
- `digits` The number of decimal places to round model selection criteria to (defaults to 3).
- `details` Logical indicating whether some additional details should be printed, defaults to `TRUE`. Exists to facilitate MoE_stepwise printing.

## Details

The purpose of this function is to conduct model selection on "MoEClust" objects, fit to the same data set, with different combinations of gating/expert network covariates or different initialisation settings.

Model selection will have already been performed in terms of choosing the optimal number of components and GPCM/**mclust** model type within each supplied set of results, but MoE_compare will respect the internal ranking of models when producing the final ranking if `optimal.only` is `FALSE`: otherwise only those models already deemed optimal within each "MoEClust" object will be ranked.

As such if two sets of results are supplied when `optimal.only` is `FALSE`, the 1st, 2nd and 3rd best models could all belong to the first set of results, meaning a model deemed suboptimal according to one set of covariates could be superior to one deemed optimal under another set of covariates.

### Value

A list of class `"MoECompare"`, for which a dedicated print function exists, containing the following elements (each of length `pick`, and ranked according to `criterion`, where appropriate):

| | |
|---|---|
| `data` | The name of the data set to which the models were fitted. |
| `optimal` | The single optimal model (an object of class `"MoEClust"`) among those supplied, according to the chosen `criterion`. |
| `pick` | The final number of ranked models. May be different (i.e. less than) the supplied `pick` value. |
| `MoENames` | The names of the supplied `"MoEClust"` objects. |
| `modelNames` | The `mclustModelNames`. |
| `G` | The optimal numbers of components. |
| `df` | The numbers of estimated parameters. |
| `iters` | The numbers of EM/CEM iterations. |
| `bic` | BIC values, ranked according to `criterion`. |
| `icl` | TCL values, ranked according to `criterion`. |
| `aic` | AIC values, ranked according to `criterion`. |
| `loglik` | Maximal log-likelihood values, ranked according to `criterion`. |
| `gating` | The gating formulas. |
| `expert` | The expert formulas. |
| `algo` | The algorithm used for fitting the model - either `"EM"`, `"CEM"`, `"cemEM"`. |
| `equalPro` | Logical indicating whether mixing proportions were constrained to be equal across components. |
| `hypvol` | Hypervolume parameters for the noise component if relevant, otherwise set to `NA` (see `MoE_control`). |
| `noise` | The type of noise component fitted (if any). Only displayed if at least one of the compared models has a noise component. |
| `noise.gate` | Logical indicating whether gating covariates were allowed to influence the noise component's mixing proportion. Only printed for models with a noise component, when at least one of the compared models has gating covariates. |
| `equalNoise` | Logical indicating whether the mixing proportion of the noise component for `equalPro` models is also equal (`TRUE`) or estimated (`FALSE`). |

### Note

The `criterion` argument here need not comply with the criterion used for model selection within each `"MoEClust"` object, but be aware that a mismatch in terms of `criterion` *may* require the optimal model to be re-fit in order to be extracted, thereby slowing down `MoE_compare`.

If random starts had been used via `init.z="random"` the `optimal` model may not necessarily correspond to the highest-ranking model in the presence of a criterion mismatch, due to the randomness of the initialisation.

A dedicated `print` function exists for objects of class `"MoECompare"`.

`plot.MoEClust` and `as.Mclust` can both also be called on objects of class `"MoECompare"`.

## Author(s)

Keefe Murphy - <<keefe.murphy@ucd.ie>>

## References

Murphy, K. and Murphy, T. B. (2019). Gaussian parsimonious clustering models with covariates and a noise component. *Advances in Data Analysis and Classification*, 1-33. <doi:10.1007/s11634-019-00373-8>.

## See Also

See `MoE_stepwise` for identifying the optimal model and its covariates via greedy forward stepwise selection.

`MoE_clust`, `mclustModelNames`, `plot.MoEClust`, `as.Mclust`

## Examples

```
data(CO2data)
CO2   <- CO2data$CO2
GNP   <- CO2data$GNP

# Fit a range of models
m1    <- MoE_clust(CO2, G=1:3)
m2    <- MoE_clust(CO2, G=2:3, gating= ~ GNP)
m3    <- MoE_clust(CO2, G=1:3, expert= ~ GNP)
m4    <- MoE_clust(CO2, G=2:3, gating= ~ GNP, expert= ~ GNP)
m5    <- MoE_clust(CO2, G=2:3, equalPro=TRUE)
m6    <- MoE_clust(CO2, G=2:3, expert= ~ GNP, equalPro=TRUE)
m7    <- MoE_clust(CO2, G=2:3, expert= ~ GNP, tau0=0.1)

# Rank only the optimal models and examine the best model
(comp <- MoE_compare(m1, m2, m3, m4, m5, m6, m7, optimal.only=TRUE))
(best <- comp$optimal)
(summ <- summary(best, classification=TRUE, parameters=TRUE, networks=TRUE))

# Examine all models visited, including those already deemed suboptimal
# Only print models with expert covariates & more than one component
comp2 <- MoE_compare(m1, m2, m3, m4, m5, m6, m7, pick=Inf)
print(comp2, comp2$expert != "None" & comp2$G > 1)

# Conduct a stepwise search on the same data
(mod1 <- MoE_stepwise(CO2, GNP))
```

```
# Conduct another stepwise search considering models with a noise component
(mod2 <- MoE_stepwise(CO2, GNP, noise=TRUE))

# Compare both sets of results to choose the optimal model
(best <- MoE_compare(mod1, mod2, optimal.only=TRUE)$optimal)
```

---

MoE_control *Set control values for use with MoEClust*

---

## Description

Supplies a list of arguments (with defaults) for use with MoE_clust.

## Usage

```
MoE_control(init.z = c("hc", "quantile", "kmeans", "mclust", "random", "list"),
            noise.args = list(...),
            equalPro = FALSE,
            exp.init = list(...),
            algo = c("EM", "CEM", "cemEM"),
            criterion = c("bic", "icl", "aic"),
            stopping = c("aitken", "relative"),
            z.list = NULL,
            nstarts = 1L,
            eps = .Machine$double.eps,
            tol = c(1e-05, sqrt(.Machine$double.eps), 1e-08),
            itmax = c(.Machine$integer.max, .Machine$integer.max, 100L),
            hc.args = list(...),
            km.args = list(...),
            init.crit = c("bic", "icl"),
            warn.it = 0L,
            MaxNWts = 1000L,
            verbose = interactive(),
            ...)
```

## Arguments

`init.z` The method used to initialise the cluster labels. Defaults to a model-based agglomerative hierarchical clustering tree as per "hc" for multivariate data (see hc.args), or "quantile"-based clustering as per quant_clust for univariate data (unless there are expert network covariates incorporated via exp.init$joint &/or exp.init$clustMD, in which case the default is again "hc"). The "quantile" option is thus only available for univariate data when expert network covariates are not incorporated via exp.init$joint &/or exp.init$clustMD, or when expert network covariates are not supplied.

Other options include "kmeans" (see km.args), "random" initialisation, a user-supplied "list", and a full run of Mclust (itself initialised via a model-based agglomerative hierarchical clustering tree, again see hc.args), although this last option "mclust" will be coerced to "hc" if there are no gating &/or expert covariates within MoE_clust (in order to better reproduce Mclust output).

When init.z="list", exp.init$clustMD is forced to FALSE; otherwise, when isTRUE(exp.init$clustMD) and the clustMD library is loaded, the init.z argument instead governs the method by which a call to clustMD is initialised. In this instance, "quantile" will instead default to "hc", and the arguments to hc.args and km.args will be ignored (unless all clustMD model types fail for a given number of components).

When init.z="mclust" or clustMD is successfully invoked (via exp.init$clustMD), the argument init.crit (see below) specifies the model-selection criterion ("bic" or "icl") by which the optimal Mclust or clustMD model type to initialise with is determined, and criterion remains unaffected.

noise.args A list supplying select named parameters to control inclusion of a noise component in the estimation of the mixture. If either or both of the arguments tau0 &/or noise.init are supplied, a noise component is added to the the model in the estimation.

- tau0 Prior mixing proportion for the noise component. If supplied, a noise component will be added to the model in the estimation, with tau0 giving the prior probability of belonging to the noise component for *all* observations. Typically supplied as a scalar in the interval (0, 1), e.g. 0.1. Can be supplied as a vector when gating covariates are present and noise.args$noise.gate is TRUE. This argument can be supplied instead of or in conjunction with the argument noise.init below.
- noise.init A logical or numeric vector indicating an initial guess as to which observations are noise in the data. If numeric, the entries should correspond to row indices of the data. If supplied, a noise component will be added to the model in the estimation. This argument can be used in conjunction with tau0 above, or can be replaced by that argument also.
- noise.gate A logical indicating whether gating network covariates influence the mixing proportion for the noise component, if any. Defaults to TRUE, but leads to greater parsimony if FALSE. Only relevant in the presence of a noise component; only effects estimation in the presence of gating covariates.
- noise.meth The method used to estimate the volume when a noise component is invoked. Defaults to hypvol. For univariate data, this argument is ignored and the range of the data is used instead (unless noise.vol below is specified). The options "convexhull" and "ellipsoidhull" require loading the geometry and cluster libraries, respectively. This argument is only relevant if noise.vol below is not supplied.
- noise.vol This argument can be used to override the argument noise.meth by specifying the (hyper)volume directly, i.e. specifying an improper uniform density. This will override the use of the range of the response data for univariate data if supplied. Note that the (hyper)volume, rather than its inverse, is supplied here. This can affect prediction and the location of the MVN ellipses for MoE_gpairs plots (see noise_vol).
- equalNoise Logical which is only invoked when isTRUE(equalPro) and gating covariates are not supplied. Under the default setting (FALSE), the mix-

ing proportion for the noise component is estimated, and remaining mixing proportions are equal; when TRUE all components, including the noise component, have equal mixing proportions.

discard.noise A logical governing how the means are summarised in parameters$mean and by extension the location of the MVN ellipses in MoE_gpairs plots for models with *both* expert network covariates and a noise component (otherwise this argument is irrelevant).
The means for models with expert network covariates are summarised by the posterior mean of the fitted values. By default (FALSE), the mean of the noise component is accounted for in the posterior mean. Otherwise, or when the mean of the noise component is unavailable (due to having been manually supplied via noise.args$noise.vol), the z matrix is renormalised after discarding the column corresponding to the noise component prior to computation of the posterior mean. The renormalisation approach can be forced by specifying noise.args$discard.noise=TRUE, even when the mean of the noise component is available. For models with a noise component fitted with algo="CEM", a small extra E-step is conducted for observations assigned to the non-noise components in this case.

In particular, the argument noise.meth will be ignored for high-dimensional n <= d data, in which case the argument noise.vol *must be* specified. Note that this forces noise.args$discard.noise to TRUE. See noise_vol for more details.

The arguments tau0 and noise.init can be used separately, to provide alternative means to invoke a noise component. However, they can also be supplied together, in which case observations corresponding to noise.init have probability tau0 (rather than 1) of belonging to the noise component.

equalPro Logical variable indicating whether or not the mixing proportions are to be constrained to be equal in the model. Default: equalPro = FALSE. Only relevant when gating covariates are *not* supplied within MoE_clust, otherwise ignored. In the presence of a noise component (see noise.args), only the mixing proportions for the non-noise components are constrained to be equal (by default, see equalNoise), after accounting for the noise component.

exp.init A list supplying select named parameters to control the initialisation routine in the presence of *expert* network covariates (otherwise ignored):

joint A logical indicating whether the initial partition is obtained on the joint distribution of the response and expert network covariates (defaults to TRUE) or just the response variables (FALSE). By default, only continuous expert network covariates are considered (see exp.init$clustMD below). Only relevant when init.z is not "random" (unless isTRUE(exp.init$clustMD), in which case init.z specifies the initialisation routine for a call to clustMD). This will render the "quantile" option to init.z for univariate data unusable if continuous expert network covariates are supplied &/or categorical/ordinal expert network covariates are supplied when isTRUE(exp.init$clustMD) and the clustMD library is loaded.

mahalanobis A logical indicating whether to iteratively reallocate observations during the initialisation phase to the component corresponding to the expert network regression to which it's closest to the fitted values of in terms of

Mahalanobis distance (defaults to `TRUE`). This will ensure that each component can be well modelled by a single expert prior to running the EM/CEM algorithm.

- `clustMD` A logical indicating whether categorical/ordinal covariates should be incorporated when using the joint distribution of the response and expert network covariates for initialisation (defaults to `FALSE`). Only relevant when `isTRUE(exp.init$joint)`. Requires the use of the `clustMD` library. Note that initialising in this manner involves fitting all `clustMD` model types in parallel for all numbers of components considered, and may fail (especially) in the presence of nominal expert network covariates.
  Unless `init.z="list"`, supplying this argument as `TRUE` when the `clustMD` library is loaded has the effect of superseding the `init.z` argument: this argument now governs instead how the call to `clustMD` is initialised (unless all `clustMD` model types fail for a given number of components, in which case `init.z` is invoked *instead* to initialise for `G` values for which all `clustMD` model types failed). Similarly, the arguments `hc.args` and `km.args` will be ignored (again, unless all `clustMD` model types fail for a given number of components).
- `max.init` The maximum number of iterations for the Mahalanobis distance-based reallocation procedure when `exp.init$mahalanobis` is `TRUE`. Defaults to `.Machine$integer.max`.
- `identity` A logical indicating whether the identity matrix (corresponding to the use of the Euclidean distance) is used in place of the covariance matrix of the residuals (corresponding to the use of the Mahalanobis distance). Defaults to `FALSE`; only relevant for multivariate response data.
- `drop.break` When `isTRUE(exp.init$mahalanobis)` observations will be completely in or out of a component during the initialisation phase. As such, it may occur that constant columns will be present when building a given component's expert regression (particularly for categorical covariates). It may also occur, due to this partitioning, that "unseen" data, when calculating the residuals, will have new factor levels. When `isTRUE(exp.init$drop.break)`, the Mahalanobis distance based initialisation phase will explicitly fail in either of these scenarios.
  Otherwise, `drop_constants` and `drop_levels` will be invoked when `exp.init$drop.break` is `FALSE` (the default) to *try* to remedy the situation. In any case, only a warning that the initialisation step failed will be printed, regardless of the value of `exp.init$drop.break`.

`algo` Switch controlling whether models are fit using the `"EM"` (the default) or `"CEM"` algorithm. The option `"cemEM"` allows running the EM algorithm starting from convergence of the CEM algorithm.

`criterion` When either `G` or `modelNames` is a vector, `criterion` determines whether the `"bic"` (Bayesian Information Criterion), `"icl"` (Integrated Complete Likelihood), `"aic"` (Akaike Information Criterion) is used to determine the 'best' model when gathering output. Note that all criteria will be returned in any case.

`stopping` The criterion used to assess convergence of the EM/CEM algorithm. The default (`"aitken"`) uses Aitken's acceleration method via `aitken`, otherwise the `"relative"` change in log-likelihood is monitored (which may be less strict).

Both stopping rules are ultimately governed by `tol[1]`. When the `"aitken"` method is employed, the asymptotic estimate of the final converged maximised log-likelihood is also returned as `linf` for models with 2 or more components, though the largest element of the returned vector `loglik` still gives the log-likelihood value achieved by the parameters returned at convergence, under both `stopping` methods (see `MoE_clust`).

`z.list`
: A user supplied list of initial cluster allocation matrices, with number of rows given by the number of observations, and numbers of columns given by the range of component numbers being considered. Only relevant if `init.z == "z.list"`. These matrices are allowed correspond to both soft or hard clusterings, and will be internally normalised so that the rows sum to 1.

`nstarts`
: The number of random initialisations to use when `init.z="random"`. Defaults to `1`. Results will be based on the random start yielding the highest estimated log-likelihood. Note that all `nstarts` random initialisations are affected by `exp.init$mahalanobis`, if invoked in the presence of expert network covariates, which may remove some of the randomness.

`eps`
: A scalar tolerance associated with deciding when to terminate computations due to computational singularity in covariances. Smaller values of `eps` allow computations to proceed nearer to singularity. The default is the relative machine precision `.Machine$double.eps`, which is approximately *2e-16* on IEEE-compliant machines.

`tol`
: A vector of length three giving relative convergence tolerances for 1) the log-likelihood of the EM/CEM algorithm, 2) parameter convergence in the inner loop for models with iterative M-step (`"VEI"`, `"EVE"`, `"VEE"`, `"VVE"`, `"VEV"`), and 3) optimisation in the multinomial logistic regression in the gating network, respectively. The default is `c(1e-05,sqrt(.Machine$double.eps),1e-08)`. If only one number is supplied, it is used as the tolerance for all three cases given.

`itmax`
: A vector of length three giving integer limits on the number of iterations for 1) the EM/CEM algorithm, 2) the inner loop for models with iterative M-step (`"VEI"`, `"EVE"`, `"VEE"`, `"VVE"`, `"VEV"`), and 3) the multinomial logistic regression in the gating network, respectively.

The default is `c(.Machine$integer.max,.Machine$integer.max,100)` allowing termination to be completely governed by `tol` for the inner and outer loops of the EM. If only one number is supplied, it is used as the iteration limit for the outer loop only.

`hc.args`
: A list supplying select named parameters to control the initialisation of the cluster allocations when `init.z="hc"` (or when `init.z="mclust"`, which itself relies on `hc`), unless `isTRUE(exp.init$clustMD)`, the `clustMD` library is loaded, and none of the `clustMD` model types fail (otherwise irrelevant):

`hcUse` A string specifying the type of input variables to be used. Unlike `Mclust`, this defaults to "VARS" here.

`hc.meth` A character string indicating the model to be used when hierarchical clustering (see `hc`) is employed for initialisation (either when `init.z="hc"` or `init.z="mclust"`). Defaults to `"EII"` for high-dimensional data, or `"VVV"` otherwise.

km.args — A list supplying select named parameters to control the initialisation of the cluster allocations when init.z="kmeans", unless isTRUE(exp.init$clustMD), the clustMD library is loaded, and none of the clustMD model types fail (otherwise irrelevant):

- kstarts The number of random initialisations to use. Defaults to 10.
- kiters The maximum number of K-Means iterations allowed. Defaults to 10.

init.crit — The criterion to be used to determine the optimal model type to initialise with, when init.z="mclust" or when isTRUE(exp.init$clustMD) and the clustMD library is loaded (one of "bic" or "icl"). Defaults to "icl" when criterion="icl", otherwise defaults to "bic". The criterion argument remains unaffected.

warn.it — A single number giving the iteration count at which a warning will be printed if the EM/CEM algorithm has failed to converge. Defaults to 0, i.e. no warning (which is true for any warn.it value less than 3), otherwise the message is printed regardless of the value of verbose. If non-zero, warn.it should be moderately large, but obviously less than itmax[1]. A warning will always be printed if one of more models fail to converge in itmax[1] iterations.

MaxNWts — The maximum allowable number of weights in the call to multinom for the multinomial logistic regression in the gating network. There is no intrinsic limit in the code, but increasing MaxNWts will probably allow fits that are very slow and time-consuming. It may be necessary to increase MaxNWts when categorical concomitant variables with many levels are included or the number of components is high.

verbose — Logical indicating whether to print messages pertaining to progress to the screen during fitting. By default is TRUE if the session is interactive, and FALSE otherwise. If FALSE, warnings and error messages will still be printed to the screen, but everything else will be suppressed.

... — Catches unused arguments.

## Details

MoE_control is provided for assigning values and defaults within MoE_clust and MoE_stepwise.

While the criterion argument controls the choice of the optimal number of components and GPCM/**mclust** model type, MoE_compare is provided for choosing between fits with different combinations of covariates or different initialisation settings.

## Value

A named list in which the names are the names of the arguments and the values are the values supplied to the arguments.

## Note

Note that successfully invoking exp.init$clustMD (though it defaults to FALSE) affects the role of the arguments init.z, hc.args, and km.args. Please read the documentation above carefully in this instance.

The initial allocation matrices before and after the invocation of the exp.init related arguments are both stored as attributes in the object returned by MoE_clust (named "Z.init" and "Exp.init", respectively). If `init.z="random"` and `nstarts` > 1, the allocations corresponding to the best random start are stored. This can be useful for supplying `z.list` for future fits.

## Author(s)

Keefe Murphy - <<keefe.murphy@ucd.ie>>

## See Also

`MoE_clust`, `MoE_stepwise`, `aitken`, `hc`, `mclust.options`, `quant_clust`, `clustMD`, `noise_vol`, `hypvol`, `convhulln`, `ellipsoidhull`, `MoE_compare`, `multinom`

## Examples

```
ctrl1 <- MoE_control(criterion="icl", itmax=100, warn.it=15, init.z="random", nstarts=5)

data(CO2data)
GNP   <- CO2data$GNP
res   <- MoE_clust(CO2data$CO2, G=2, expert = ~ GNP, control=ctrl1)

# Alternatively, specify control arguments directly
res2  <- MoE_clust(CO2data$CO2, G=2, expert = ~ GNP, stopping="relative")

# Supplying ctrl1 without naming it as 'control' can throw an error
## Not run:
res3  <- MoE_clust(CO2data$CO2, G=2, expert = ~ GNP, ctrl1)
## End(Not run)

# Similarly, supplying control arguments via a mix of the ... construct
# and the named argument 'control' also throws an error
## Not run:
res4  <- MoE_clust(CO2data$CO2, G=2, expert = ~ GNP, control=ctrl1, init.z="kmeans")
## End(Not run)

# Initialise via the mixed-type joint distribution of response & covariates
# Let the ICL criterion determine the optimal clustMD model type
# Constrain the mixing proportions to be equal
ctrl2 <- MoE_control(exp.init=list(clustMD=TRUE), init.crit="icl", equalPro=TRUE)
data(ais)
library(clustMD)
res4  <- MoE_clust(ais[,3:7], G=2, modelNames="EVE", expert=~sex,
                   network.data=ais, control=ctrl2)

# Include a noise component by specifying its prior mixing proportion
res5  <- MoE_clust(ais[,3:7], G=2, modelNames="EVE", expert=~sex,
                   network.data=ais, tau0=0.1)
```

MoE_crit *MoEClust BIC, ICL, and AIC Model-Selection Criteria*

## Description

Computes the BIC (Bayesian Information Criterion), ICL (Integrated Complete Likelihood), and AIC (Akaike Information Criterion) for parsimonious mixture of experts models given the log-likelihood, the dimension of the data, the number of mixture components in the model, the numbers of parameters in the gating and expert networks respectively, and, for the ICL, the numbers of observations in each component.

## Usage

```
MoE_crit(modelName,
         loglik,
         n,
         d,
         G,
         gating.pen = G - 1L,
         expert.pen = G * d,
         z = NULL,
         df = NULL)
```

## Arguments

| | |
|---|---|
| `modelName` | A character string indicating the model. The help file for `mclustModelNames` describes the available models. |
| `loglik` | The log-likelihood for a data set with respect to the Gaussian mixture model specified in the `modelName` argument. |
| `n, d, G` | The number of observations in the data, dimension of the data, and number of components in the Gaussian mixture model, respectively, used to compute `loglik`. `d` & `G` are not necessary if `df` is supplied. |
| `gating.pen` | The number of parameters of the *gating* network of the MoEClust model. Defaults to `G -1`, which corresponds to no gating covariates. If covariates are included, this should be the number of regression coefficients in the fitted `gating` object. If there are no covariates and mixing proportions are further assumed to be present in equal proportion, `gating.pen` should be `0`. The number of parameters used in the estimation of the noise component, if any, should also be included. Not necessary if `df` is supplied. |
| `expert.pen` | The number of parameters of the *expert* network of the MoEClust model. Defaults to `G * d`, which corresponds to no expert covariates. If covariates are included, this should be the number of regression coefficients in the fitted `expert` object. Not necessary if `df` is supplied. |
| `z` | The `n` times `G` responsibility matrix whose `[i,k]`-th entry is the probability that observation *i* belonds to the *k*-th component.. If supplied the ICL is also computed and returned, otherwise only the BIC and AIC. |

df
: An alternative way to specify the number of estimated parameters (or 'used' degrees of freedom) exactly. If supplied, the arguments d,G,gating.pen and expert.pen, which are used to calculate the number of parameters, will be ignored. The number of parameters used in the estimation of the noise component, if any, should also be included.

## Details

The function is vectorised with respect to the arguments modelName and loglik.

If model is an object of class "MoEClust" with G components, the number of parameters for the gating.pen and expert.pen are length(coef(model$gating)) and G * length(coef(model$expert[[1]])), respectively.

Models with a noise component are facilitated here too provided the extra number of parameters are accounted for by the user.

## Value

A simplified array containing the BIC, AIC, number of estimated parameters (df) and, if z is supplied, also the ICL, for each of the given input arguments.

## Note

In order to speed up repeated calls to the function inside MoE_clust, no checks take place.

## Author(s)

Keefe Murphy - <<keefe.murphy@ucd.ie>>

## References

Biernacki, C., Celeux, G. and Govaert, G. (2000). Assessing a mixture model for clustering with the integrated completed likelihood. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 22(7): 719-725.

## See Also

MoE_clust, nVarParams, mclustModelNames

## Examples

```
MoE_crit(modelName=c("VVI", "VVE", "VVV"), n=120, d=8,
         G=3, loglik=c(-4036.99, -3987.12, -3992.45))

data(CO2data)
GNP   <- CO2data$GNP
model <- MoE_clust(CO2data$CO2, G=1:2, expert= ~ GNP)
G     <- model$G
name  <- model$modelName
ll    <- max(model$loglik)
n     <- length(CO2data$CO2)
```

```
z     <- model$z

# Compare BIC from MoE_crit to the BIC of the model
(bic2 <- MoE_crit(modelName=name, loglik=ll, n=n, d=1, G=G, z=z,
                  expert.pen=G * length(coef(model$expert[[1]])))["bic",])
identical(bic2, unname(model$bic)) #TRUE

# Make the same comparison with the known number of estimated parameters
(bic3 <- MoE_crit(modelName=name, loglik=ll, n=n, G=G, df=model$df, z=z)["bic",])
identical(bic3, bic2)              #TRUE
```

---

MoE_cstep *C-step for MoEClust Models*

---

## Description

Function to compute the assignment matrix z and the conditional log-likelihood for MoEClust models, with the aid of MoE_dens.

## Usage

```
MoE_cstep(data,
          mus,
          sigs,
          log.tau = 0L,
          Vinv = NULL,
          Dens = NULL)
```

## Arguments

| | |
|---|---|
| data | If there are no expert network covariates, `data` should be a numeric matrix or data frame, wherein rows correspond to observations (n) and columns correspond to variables (d). If there are expert network covariates, this should be a list of length G containing matrices/data.frames of (multivariate) WLS residuals for each component. |
| mus | The mean for each of G components. If there is more than one component, this is a matrix whose k-th column is the mean of the k-th component of the mixture model. For the univariate models, this is a G-vector of means. In the presence of expert network covariates, all values should be equal to `0`. |
| sigs | The `variance` component in the parameters list from the output to eg. MoE_clust. The components of this list depend on the specification of `modelName` (see mclustVariance for details). The number of components `G`, the number of variables `d`, and the `modelName` are inferred from `sigs`. |
| log.tau | If covariates enter the gating network, an n times G matrix of mixing proportions, otherwise a G-vector of mixing proportions for the components of the mixture. **Must** be on the log-scale in both cases. The default of `0` effectively means densities (or log-densities) aren't scaled by the mixing proportions. |

`Vinv` An estimate of the reciprocal hypervolume of the data region. See the function `noise_vol`. Used only if an initial guess as to which observations are noise is supplied. Mixing proportion(s) must be included for the noise component also.

`Dens` (Optional) A numeric matrix whose `[i,k]`-th entry is the **log**-density of observation $i$ in component $k$, scaled by the mixing proportions, to which the function is to be applied, typically obtained by `MoE_dens` but this is not necessary. If this is supplied, all other arguments are ignored, otherwise `MoE_dens` is called according to the other supplied arguments.

## Value

A list containing two elements:

`z` A matrix with `n` rows and `G` columns containing 1 where the observation belongs to the cluster indicated by the column number, and 0 otherwise.

`loglik` The estimated conditional log-likelihood.

## Note

This function is intended for joint use with `MoE_dens`, using the **log**-densities. Caution is advised using this function without explicitly naming the arguments. Models with a noise component are facilitated here too.

The C-step can be replaced by an E-step, see `MoE_estep` and the `algo` argument to `MoE_control`.

## Author(s)

Keefe Murphy - <`<keefe.murphy@ucd.ie>`>

## See Also

`MoE_dens`, `MoE_clust`, `MoE_estep`, `MoE_control`, `mclustVariance`

## Examples

```
# MoE_cstep can be invoked for fitting MoEClust models via the CEM algorithm
# via the 'algo' argument to MoE_control:
data(ais)
hema   <- ais[,3:7]
model  <- MoE_clust(hema, G=3, gating= ~ BMI + sex, modelNames="EEE", network.data=ais, algo="CEM")
Dens   <- MoE_dens(data=hema, mus=model$parameters$mean,
                   sigs=model$parameters$variance, log.tau=log(model$parameters$pro))

# Construct the z matrix and compute the conditional log-likelihood
Cstep  <- MoE_cstep(Dens=Dens)
(ll    <- Cstep$loglik)

# Check that the z matrix & classification are the same as those from the model
identical(max.col(Cstep$z), as.integer(unname(model$classification))) #TRUE
identical(Cstep$z, model$z)                                            #TRUE
```

```
# Call MoE_cstep directly
Cstep2 <- MoE_cstep(data=hema, sigs=model$parameters$variance,
                    mus=model$parameters$mean, log.tau=log(model$parameters$pro))
identical(Cstep2$loglik, ll)                                         #TRUE
```

---

MoE_dens *Density for MoEClust Mixture Models*

---

## Description

Computes densities (or log-densities) of observations in MoEClust mixture models.

## Usage

```
MoE_dens(data,
         mus,
         sigs,
         log.tau = 0L,
         Vinv = NULL,
         logarithm = TRUE)
```

## Arguments

| | |
|---|---|
| data | If there are no expert network covariates, `data` should be a numeric matrix or data frame, wherein rows correspond to observations (n) and columns correspond to variables (d). If there are expert network covariates, this should be a list of length G containing matrices/data.frames of (multivariate) WLS residuals for each component. |
| mus | The mean for each of G components. If there is more than one component, this is a matrix whose k-th column is the mean of the k-th component of the mixture model. For the univariate models, this is a G-vector of means. In the presence of expert network covariates, all values should be equal to `0`. |
| sigs | The `variance` component in the parameters list from the output to eg. `MoE_clust`. The components of this list depend on the specification of `modelName` (see `mclustVariance` for details). The number of components `G`, the number of variables `d`, and the `modelName` are inferred from `sigs`. |
| log.tau | If covariates enter the gating network, an n times G matrix of mixing proportions, otherwise a G-vector of mixing proportions for the components of the mixture. **Must** be on the log-scale in both cases. The default of `0` effectively means densities (or log-densities) aren't scaled by the mixing proportions. |
| Vinv | An estimate of the reciprocal hypervolume of the data region. See the function `noise_vol`. Used only if an initial guess as to which observations are noise is supplied. Mixing proportion(s) must be included for the noise component also. |
| logarithm | A logical value indicating whether or not the logarithm of the component densities should be returned. This defaults to `TRUE`, otherwise component densities are returned, obtained from the component log-densities by exponentiation. The **log**-densities can be passed to `MoE_estep` or `MoE_cstep`. |

### Value

A numeric matrix whose `[i,k]`-th entry is the density or log-density of observation *i* in component *k*, scaled by the mixing proportions. These densities are unnormalised.

### Note

This function is intended for joint use with `MoE_estep` or `MoE_cstep`, using the **log**-densities. Note that models with a noise component are facilitated here too.

### Author(s)

Keefe Murphy - <`<keefe.murphy@ucd.ie>`>

### See Also

`MoE_estep`, `MoE_cstep`, `MoE_clust`, `mclustVariance`

### Examples

```
data(ais)
hema  <- ais[,3:7]
model <- MoE_clust(hema, G=3, gating= ~ BMI + sex, modelNames="EEE", network.data=ais)
Dens  <- MoE_dens(data=hema, mus=model$parameters$mean,
                  sigs=model$parameters$variance, log.tau=log(model$parameters$pro))

# Construct the z matrix and compute the log-likelihood
Estep <- MoE_estep(Dens=Dens)
(ll   <- Estep$loglik)

# Check that the z matrix & classification are the same as those from the model
identical(max.col(Estep$z), as.integer(unname(model$classification))) #TRUE
identical(Estep$z, model$z)                                             #TRUE

# The same can be done for models with expert covariates &/or a noise component
# Note for models with expert covariates that the mean has to be supplied as 0
m2    <- MoE_clust(hema, G=2, expert= ~ sex, modelNames="EVE", network.data=ais, tau0=0.1)
Dens2 <- MoE_dens(data=m2$resid.data, sigs=m2$parameters$variance, mus=0,
                  log.tau=log(m2$parameters$pro), Vinv=m2$parameters$Vinv)
```

---

## MoE_estep — *E-step for MoEClust Models*

---

### Description

Softmax function to compute the responsibility matrix z and the log-likelihood for MoEClust models, with the aid of `MoE_dens`.

## Usage

```
MoE_estep(data,
          mus,
          sigs,
          log.tau = 0L,
          Vinv = NULL,
          Dens = NULL)
```

## Arguments

| | |
|---|---|
| data | If there are no expert network covariates, data should be a numeric matrix or data frame, wherein rows correspond to observations (n) and columns correspond to variables (d). If there are expert network covariates, this should be a list of length G containing matrices/data.frames of (multivariate) WLS residuals for each component. |
| mus | The mean for each of G components. If there is more than one component, this is a matrix whose k-th column is the mean of the k-th component of the mixture model. For the univariate models, this is a G-vector of means. In the presence of expert network covariates, all values should be equal to 0. |
| sigs | The variance component in the parameters list from the output to eg. MoE_clust. The components of this list depend on the specification of modelName (see mclustVariance for details). The number of components G, the number of variables d, and the modelName are inferred from sigs. |
| log.tau | If covariates enter the gating network, an n times G matrix of mixing proportions, otherwise a G-vector of mixing proportions for the components of the mixture. **Must** be on the log-scale in both cases. The default of 0 effectively means densities (or log-densities) aren't scaled by the mixing proportions. |
| Vinv | An estimate of the reciprocal hypervolume of the data region. See the function noise_vol. Used only if an initial guess as to which observations are noise is supplied. Mixing proportion(s) must be included for the noise component also. |
| Dens | (Optional) A numeric matrix whose [i,k]-th entry is the **log**-density of observation *i* in component *k*, scaled by the mixing proportions, to which the softmax function is to be applied, typically obtained by MoE_dens but this is not necessary. If this is supplied, all other arguments are ignored, otherwise MoE_dens is called according to the other supplied arguments. |

## Value

A list containing two elements:

| | |
|---|---|
| z | A matrix with n rows and G columns containing the probability of cluster membership for each of n observations and G clusters. |
| loglik | The estimated log-likelihood, computed efficiently via rowLogSumExps. |

## Note

This softmax function is intended for joint use with MoE_dens, using the **log**-densities. Caution is advised using this function without explicitly naming the arguments. Models with a noise component are facilitated here too.

The E-step can be replaced by a C-step, see MoE_cstep and the algo argument to MoE_control.

## Author(s)

Keefe Murphy - <<keefe.murphy@ucd.ie>>

## See Also

MoE_dens, MoE_clust, MoE_cstep, MoE_control, mclustVariance, rowLogSumExps

## Examples

```
data(ais)
hema   <- ais[,3:7]
model  <- MoE_clust(hema, G=3, gating= ~ BMI + sex, modelNames="EEE", network.data=ais)
Dens   <- MoE_dens(data=hema, mus=model$parameters$mean,
                   sigs=model$parameters$variance, log.tau=log(model$parameters$pro))

# Construct the z matrix and compute the log-likelihood
Estep  <- MoE_estep(Dens=Dens)
(ll    <- Estep$loglik)

# Check that the z matrix & classification are the same as those from the model
identical(max.col(Estep$z), as.integer(unname(model$classification))) #TRUE
identical(Estep$z, model$z)                                            #TRUE

# Call MoE_estep directly
Estep2 <- MoE_estep(data=hema, sigs=model$parameters$variance,
                    mus=model$parameters$mean, log.tau=log(model$parameters$pro))
identical(Estep2$loglik, ll)                                           #TRUE

# The same can be done for models with expert covariates &/or a noise component
# Note for models with expert covariates that the mean has to be supplied as 0
m2    <- MoE_clust(hema, G=2, expert= ~ sex, modelNames="EVE", network.data=ais, tau0=0.1)
Estep3 <- MoE_estep(data=m2$resid.data, sigs=m2$parameters$variance, mus=0,
                    log.tau=log(m2$parameters$pro), Vinv=m2$parameters$Vinv)
```

---

MoE_gpairs *Generalised Pairs Plots for MoEClust Mixture Models*

---

## Description

Produces a matrix of plots showing pairwise relationships between continuous response variables and continuous/categorical/logical/ordinal associated covariates, as well as the clustering achieved, according to fitted MoEClust mixture models.

### Usage

```
MoE_gpairs(res,
           response.type = c("points", "uncertainty", "density"),
           subset = list(...),
           scatter.type = c("lm", "points"),
           conditional = c("stripplot", "boxplot"),
           addEllipses = c("outer", "yes", "no", "inner", "both"),
           expert.covar = TRUE,
           border.col = c("purple", "black", "brown", "brown", "navy"),
       bg.col = c("cornsilk", "white", "palegoldenrod", "palegoldenrod", "cornsilk"),
           outer.margins = list(bottom = grid::unit(2, "lines"),
                                left = grid::unit(2, "lines"),
                                top = grid::unit(2, "lines"),
                                right = grid::unit(2, "lines")),
           outer.labels = NULL,
           outer.rot = c(0, 90),
           gap = 0.05,
           buffer = 0.025,
           uncert.cov = FALSE,
           scatter.pars = list(...),
           density.pars = list(...),
           stripplot.pars = list(...),
           barcode.pars = list(...),
           mosaic.pars = list(...),
           axis.pars = list(...),
           diag.pars = list(...),
           ...)
```

### Arguments

`res` An object of class "MoEClust" generated by MoE_clust, or an object of class "MoECompare" generated by MoE_compare. Models with a noise component are facilitated here too.

`response.type` The type of plot desired for the scatter plots comparing continuous response variables. Defaults to "points".

Points can also be sized according to their associated clustering uncertainty with the option "uncertainty". In so doing, the transparency of the points will also be proportional to their clustering uncertainty, provided the device supports transparency. See also MoE_Uncertainty for an alternative means of visualising observation-specific cluster uncertainties (especially for univariate data).

Alternatively, the bivariate "density" contours can be displayed (see density.pars), provided there is at least one Gaussian component in the model. Caution is advised when producing density plots for models with covariates in the expert network; the required number of evaluations of the (multivariate) Gaussian density for each panel (res$G * prod(density.pars$grid.size)) increases by a factor of res$n, thus plotting may be slow (particularly for large data sets).

`subset` A list giving named arguments for producing only a subset of panels:

show.map Logical indicating whether to show panels involving the MAP classification (defaults to TRUE, unless there is only one component, in which case the MAP classification is never plotted.).

data.ind For subsetting response variables: a vector of column indices corresponding to the variables in the columns of res$data which should be shown. Defaults to all. Can be 0, in order to suppress plotting the response variables.

cov.ind For subsetting covariates: a vector of column indices corresponding to the covariates in the columns res$net.covs which should be shown. Defaults to all. Can be 0, in order to suppress plotting the covariates.

The subsetting must include at least two variables, whether they be the MAP, a response variable, or a covariate, in order to be valid for plotting purposes. The arguments data.ind and cov.ind can also be used to simply reorder the panels, without actually subsetting.

scatter.type A vector of length 2 (or 1) giving the plot type for the upper and lower triangular portions of the plot, respectively, pertaining to the associated covariates. Defaults to "lm" for covariate vs. response panels and "points" otherwise. Only relevant for models with continuous covariates in the gating &/or expert network. "ci" and "lm" type plots are only produced for plots pairing covariates with response, and never response vs. response or covariate vs. covariate. Note that lines &/or confidence intervals will only be drawn for continuous covariates included in the expert network; to include covariates included only in the gating network also, the options "lm2" or "ci2" can be used but this is not generally advisable.

conditional A vector of length 2 (or 1) giving the plot type for the upper and lower triangular portions of the plot, respectively, for plots involving a mix of categorical and continuous variables. Defaults to "stripplot" in the upper triangle and "boxplot" in the lower triangle (see panel.stripplot and panel.bwplot). "barcode" and "violin" plots can also be produced. Only relevant for models with categorical covariates in the gating &/or expert network. Comparisons of two categorical variables (which can only ever be covariates) are always displayed via mosaic plots (see strucplot).

addEllipses Controls whether to add MVN ellipses with axes corresponding to the within-cluster covariances for the response data ("yes" or "no"). The options "inner" and "outer" (the default) will colour the axes or the perimeter of those ellipses, respectively, according to the cluster they represent (according to scatter.pars$lci.col). The option "both" will obviously colour both the axes and the perimeter. Ellipses are only ever drawn for multivariate data, and only when response.type is "points" or "uncertainty".

Ellipses are centered on the posterior mean of the fitted values when there are expert network covariates, otherwise on the posterior mean of the response variables. In the presence of expert network covariates, the component-specific covariance matrices are also (by default, via the argument expert.covar below) modified for plotting purposes via the function expert_covar, in order to account for the extra variability of the means, usually resulting in bigger shapes & sizes for the MVN ellipses.

expert.covar — Logical (defaults to TRUE) governing whether the extra variability in the component means is added to the MVN ellipses corresponding to the component covariance matrices in the presence of expert network covariates when addEllipses is invoked accordingly. See the function expert_covar. Only relevant when response.type is "points" or "uncertainty".

border.col — A vector of length 5 (or 1) containing *border* colours for plots against the MAP classification, response vs. response, covariate vs. response, response vs. covariate, and covariate vs. covariate panels, respectively.

Defaults to c("purple","black","brown","brown","navy").

bg.col — A vector of length 5 (or 1) containing *background* colours for plots against the MAP classification, response vs. response, covariate vs. response, response vs. covariate, and covariate vs. covariate panels, respectively.

Defaults to c("cornsilk","white","palegoldenrod","palegoldenrod","cornsilk").

outer.margins — A list of length 4 with units as components named bottom, left, top, and right, giving the outer margins; the defaults uses two lines of text. A vector of length 4 with units (ordered properly) will work, as will a vector of length 4 with numeric variables (interpreted as lines).

outer.labels — The default is NULL, for alternating labels around the perimeter. If "all", all labels are printed, and if "none", no labels are printed.

outer.rot — A 2-vector (x, y) rotating the top/bottom outer labels x degrees and the left/right outer labels y degrees. Only works for categorical labels of boxplot and mosaic panels. Defaults to c(0,90).

gap — The gap between the tiles; defaulting to 0.05 of the width of a tile.

buffer — The fraction by which to expand the range of quantitative variables to provide plots that will not truncate plotting symbols. Defaults to 0.025, i.e. 2.5 percent of the range.

uncert.cov — A logical indicating whether the expansion factor for points on plots involving covariates should also be modified when response.type="uncertainty". Defaults to FALSE, and only relevant for scatterplot and stripplot panels.

scatter.pars — A list supplying select parameters for the continuous vs. continuous scatter plots.

NULL is equivalent to:

```
list(scat.pch=if(response.type == "uncertainty") 19 else res$classification,
scat.size=unit(0.25, "char"), scat.col=res$classification,
lci.col=res$classification, noise.size=unit(0.2, "char")),
```

where lci.col gives the colour of the fitted lines &/or confidence intervals when scatter.type is one of "ci" or "lm" and the colour of the ellipses when addEllipses is one of "outer", "inner", or "both". Note that scatter.pars$scat.size will be modified on an observation by observation level when response.type is "uncertainty". Note also that the default for scatter.pars$scat.pch changes depending on whether response.type is given as "points" or "uncertainty", though it can of course be modified in both cases. Finally, scatter.pars$noise.size can be used to modify scatter.pars$scat.size for observations assigned to the noise component (if any), but only when response.type="points".

density.pars
: A list supplying select parameters for visualising the bivariate density contours, only when `response.type` is `"density"`.

  `NULL` is equivalent to:

  `list(grid.size=c(100, 100), dcol="grey50", nlevels=11, show.labels=TRUE, label.style="mixed"),`

  where `density.pars$grid.size` is a vector of length two giving the number of points in the x & y direction of the grid over which the density is evaluated, respectively, and `density.pars$dcol` is either a single colour or a vector of length `density.pars$nlevels` colours (although note that `density.pars$dcol`, when *not* specified, will be adjusted for transparency). Finally, `density.pars$label.style` can take the values `"mixed"`, `"flat"`, or `"align"`.

stripplot.pars
: A list supplying select parameters for continuous vs. categorical panels when one of the entries of `conditional` is `"stripplot"`.

  `NULL` is equivalent to:

  `list(strip.pch=res$classification, strip.size=unit(0.5, "char"), strip.col=res$classification, jitter=TRUE, size.noise=unit(0.4, "char")),`

  where `stripplot.pars$strip.size` and `stripplot.pars$size.noise` retain the definitions for the similar arguments under `scatter.pars` above. However, `stripplot.pars$noise.size` is invoked regardless of the `response.type`.

barcode.pars
: A list supplying select parameters for continuous vs. categorical panels when one of the entries of `conditional` is `"barcode"`. See the help file for `barcode::barcode`.

  `NULL` is equivalent to:

  `list(bar.col=res$classification, nint=0, ptsize=unit(0.25, "char"), ptpch=1, bcspace=NULL, use.points=FALSE),`

  where `barcode.pars$bar.col` is only invoked for panels where the categorical variable is the MAP classification (i.e. when `isTRUE(subset$show.map)`) if it is of length greater than 1, otherwise it is used for all relevant panels. See `diag.pars$hist.color` for controlling the colours of non-MAP-related barcode panels.

mosaic.pars
: A list supplying select parameters for categorical vs. categorical panels. `NULL`. Currently `shade,gp_labels,gp`, and `gp_args` are passed through to `strucplot` for producing mosaic tiles.

axis.pars
: A list supplying select parameters for controlling axes.

  `NULL` is equivalent to:

  `list(n.ticks=5, axis.fontsize=9).`

  The argument `n.ticks` will be overwritten for categorical variables with fewer than 5 levels.

diag.pars
: A list supplying select parameters for panels along the diagonal.

  `NULL` is equivalent to:

  `list(diag.fontsize=9, show.hist=TRUE, diagonal=TRUE, hist.color=hist.color, show.counts=TRUE),` where `hist.color` is a vector of length 4, giving the colours for the response variables, gating covariates, expert covariates, and covariates entering both networks, respectively. `hist.color` also governs the fill colour for boxplot panels except those involving covariates only, as well as the colour of barcode panels not related to the MAP classification. By default, response variables are `"black"` and covariates of any kind are `"dimgrey"`. The MAP classification is always coloured by cluster membership. `show.counts` is only relevant for categorical variables.

When `diagonal=TRUE` (the default), the diagonal from the top left to the bottom right is used for displaying the marginal distributions of variables. Specifying `diagonal=FALSE` will place the diagonal running from the top right down to the bottom left.

`...` Catches unused arguments. Alternatively, named arguments can be passed directly here to any/all of `scatter.pars`, `barcode.pars`, `mosaic.pars`, `axis.pars` and `diag.pars`.

### Value

A generalised pairs plot showing all pairwise relationships between clustered response variables and associated gating &/or expert network continuous &/or categorical variables, coloured according to the MAP classification, with the marginal distributions of each variable along the diagonal.

### Note

For `MoEClust` models with more than one associated covariate (entering either network), fitted lines produced in continuous covariate vs. continuous response scatter plots via `scatter.type="lm"` or `scatter.type="ci"` will **NOT** correspond to the coefficients in the expert network (`res$expert`).

`plot.MoEClust` is a wrapper to `MoE_gpairs` which accepts the default arguments, and also produces other types of plots. Caution is advised producing generalised pairs plots when the dimension of the data is large.

### Author(s)

Keefe Murphy - <`<keefe.murphy@ucd.ie>`>

### References

Murphy, K. and Murphy, T. B. (2019). Gaussian parsimonious clustering models with covariates and a noise component. *Advances in Data Analysis and Classification*, 1-33. <doi:10.1007/s11634-019-00373-8>.

Emerson, J. W., Green, W. A., Schloerke, B., Crowley, J., Cook, D., Hofmann, H. and Wickham, H. (2013). The generalized pairs plot. *Journal of Computational and Graphical Statistics*, 22(1):79-91.

### See Also

`MoE_clust`, `MoE_stepwise`, `plot.MoEClust`, `MoE_Uncertainty`, `expert_covar`, `panel.stripplot`, `panel.bwplot`, `panel.violin`, `strucplot`

### Examples

```
data(ais)
res   <- MoE_clust(ais[,3:7], G=2, gating= ~ BMI, expert= ~ sex,
                   network.data=ais, modelNames="EVE")
MoE_gpairs(res)

# Produce the same plot, but with a violin plot in the lower triangle.
# Add fitted lines to the scatter plots.
# Size points in the response vs. response panels by their clustering uncertainty.
MoE_gpairs(res, conditional=c("stripplot", "violin"),
           scatter.type=c("lm2", "points"), response.type="uncertainty")

# Instead show the bivariate density contours of the reponse variables (without labels).
# (Plotting may be slow when response.type="density" for models with expert covariates.)
# Use different colours for histograms of covariates in the gating/expert/both networks.
# Also use different colours for response vs. covariate & covariate vs. response panels.
MoE_gpairs(res, response.type="density", show.labels=FALSE,
           hist.color=c("black", "cyan", "hotpink", "chartreuse"),
           bg.col=c("whitesmoke", "white", "mintcream", "mintcream", "floralwhite"))

# Produce a generalised pairs plot for a model with a noise component.
# Reorder the covariates and omit the variabes "Hc" and "Hg".
# Use barcode plots for the categorical/continuous pairs.
# Magnify the size of scatter points assigned to the noise component.
resN  <- MoE_clust(ais[,3:7], G=2, gating=~SSF + Ht, expert=~sex,
                   network.data=ais, modelNames="EEE", tau0=0.1, noise.gate=FALSE)
MoE_gpairs(resN, data.ind=c(1,2,5), cov.ind=c(3,1,2),
conditional="barcode", noise.size=grid::unit(0.5, "char"))
```

---

## MoE_mahala *Mahalanobis Distance Outlier Detection for Multivariate Response*

---

### Description

Computes the Mahalanobis distance between the response variable(s) and the fitted values of linear regression models with multivariate or univariate responses.

### Usage

```
MoE_mahala(fit,
           resids,
           squared = FALSE,
           identity = FALSE)
```

### Arguments

| | |
|---|---|
| `fit` | A fitted `lm` model, inheriting either the `"mlm"` or `"lm"` class. |

resids The residuals. Can be residuals for observations included in the model, or residuals arising from predictions on unseen data. Must be coercible to a matrix with the number of columns being the number of response variables. Missing values are not allowed.

squared A logical. By default (FALSE), the generalized interpoint distance is computed. Set this flag to TRUE for the squared value.

identity A logical indicating whether the identity matrix is used in in place of the precision matrix in the Mahalanobis distance calculation. Defaults to FALSE; TRUE corresponds to the use of the Euclidean distance. Only relevant for multivariate response data.

## Value

A vector giving the Mahalanobis distance (or squared Mahalanobis distance) between response(s) and fitted values for each observation.

## Author(s)

Keefe Murphy - <<keefe.murphy@ucd.ie>>

## Examples

```
data(ais)
hema <- as.matrix(ais[,3:7])
mod  <- lm(hema ~ sex + BMI, data=ais)
res  <- hema - predict(mod)
MoE_mahala(mod, res, squared=TRUE)

data(CO2data)
CO2  <- CO2data$CO2
GNP  <- CO2data$GNP
mod2 <- lm(CO2 ~ GNP, data=CO2data)
pred <- predict(mod2)
res2 <- CO2 - pred
maha <- MoE_mahala(mod2, res2)

# Highlight outlying observations
plot(GNP, CO2, type="n", ylab=expression('CO'[2]))
lines(GNP, pred, col="red")
points(GNP, CO2, cex=maha, lwd=2)
text(GNP, CO2, col="blue",
     labels=replace(as.character(CO2data$country), maha < 1, ""))

# Replicate initialisation strategy using 2 randomly chosen components
# Repeat the random initialisation if necessary
# (until 'crit' at convergence is minimised)
G       <- 3L
z       <- sample(seq_len(G), nrow(CO2data), replace=TRUE)
old     <- Inf
crit    <- .Machine$double.xmax
```

```
while(crit < old)   {
  Sys.sleep(1)
  old   <- crit
  maha  <- NULL
  plot(GNP, CO2, type="n", ylab=expression('CO'[2]))
  for(g in seq_len(G)) {
   ind  <- which(z == g)
   mod  <- lm(CO2 ~ GNP, data=CO2data, sub=ind)
   pred <- predict(mod, newdata=CO2data[,"CO2", drop=FALSE])
   maha <- cbind(maha, MoE_mahala(mod, CO2 - pred))
   lines(GNP, pred, col=g + 1L)
  }
  min.M <- rowMins(maha)
  crit  <- sum(min.M)
  z     <- max.col(maha == min.M)
  points(GNP, CO2, cex=min.M, lwd=2, col=z + 1L)
  text(GNP, CO2, col=z + 1L,
       labels=replace(as.character(CO2data$country), which(min.M <= 1), ""))
}
crit
```

---

MoE_news *Show the NEWS file*

---

## Description

Show the NEWS file of the MoEClust package.

## Usage

```
MoE_news()
```

## Value

The MoEClust NEWS file, provided the session is interactive.

## Examples

```
MoE_news()
```

MoE_plotCrit *Model Selection Criteria Plot for MoEClust Mixture Models*

## Description

Plots the BIC, ICL, AIC, or log-likelihood values of a fitted `MoEClust` object.

## Usage

```
MoE_plotCrit(res,
             criterion = c("bic", "icl", "aic", "loglik"),
             ...)
```

## Arguments

| | |
|---|---|
| `res` | An object of class `"MoEClust"` generated by `MoE_clust`, or an object of class `"MoECompare"` generated by `MoE_compare`. Models with a noise component are facilitated here too. |
| `criterion` | The criterion to be plotted. Defaults to `"bic"`. |
| `...` | Catches other arguments, or additional arguments to be passed to `plot.mclustBIC` (or equivalent functions for the other `criterion` arguments). In particular, the argument `legendArgs` to `plot.mclustBIC` can be passed. |

## Value

A plot of the values of the chosen `criterion`. The values themselves can also be returned invisibly.

## Note

`plot.MoEClust` is a wrapper to `MoE_plotCrit` which accepts the default arguments, and also produces other types of plots.

## Author(s)

Keefe Murphy - <<keefe.murphy@ucd.ie>>

## See Also

`MoE_clust`, `plot.MoEClust`, `plot.mclustBIC`

## Examples

```
# data(ais)
# res   <- MoE_clust(ais[,3:7], expert= ~ sex, network.data=ais)
# (crit <- MoE_plotCrit(res))
```

MoE_plotGate *Plot MoEClust Gating Network*

## Description

Plots the gating network for fitted MoEClust models, i.e. the observation index against the mixing proportions for that observation, coloured by cluster.

## Usage

```
MoE_plotGate(res,
             x.axis = NULL,
             type = "b",
             pch = 1,
             xlab = "Observation",
             ylab = expression(widehat(tau)[g]),
             ylim = c(0, 1),
             col = NULL,
             ...)
```

## Arguments

| | |
|---|---|
| `res` | An object of class `"MoEClust"` generated by `MoE_clust`, or an object of class `"MoECompare"` generated by `MoE_compare`. Models with a noise component are facilitated here too. |
| `x.axis` | Optional argument for the x-axis against which the mixing proportions are plotted. Defaults to `1:res$n` if missing. Supplying `x.axis` changes the defaults for the `type` and `xlab` arguments. Users are advised to only use quantities related to the gating network of the fitted model here. Furthermore, use of the `x.axis` argument is not recommended for models with more than one gating network covariate. |
| `type, pch, xlab, ylab, ylim, col` | These graphical parameters retain their definitions from `matplot`. `col` defaults to the settings in `mclust.options`. |
| `...` | Catches unused arguments, or additional arguments to be passed to `matplot`. |

## Value

A plot of the gating network of the fitted MoEClust model. The parameters of the gating network can also be returned invisibly.

## Note

`plot.MoEClust` is a wrapper to `MoE_plotGate` which accepts the default arguments, and also produces other types of plots.

By default, the noise component (if any) will be coloured `"darkgrey"`.

### Author(s)

Keefe Murphy - <<keefe.murphy@ucd.ie>>

### See Also

MoE_clust, plot.MoEClust, matplot

### Examples

```
data(ais)
res   <- MoE_clust(ais[,3:7], gating= ~ BMI, G=3, modelNames="EEE",
                   network.data=ais, noise.gate=FALSE, tau0=0.1)

# Plot against the observation index and examine the gating network coefficients
(gate <- MoE_plotGate(res))

# Plot against BMI
MoE_plotGate(res, x.axis=ais$BMI, xlab="BMI")

# Plot against a categorical covariate
res2  <- MoE_clust(ais[,3:7], gating= ~ sex, G=3, modelNames="EVE", network.data=ais)
MoE_plotGate(res2, x.axis=ais$sex, xlab="sex")
```

---

## MoE_plotLogLik *Plot the Log-Likelihood of a MoEClust Mixture Model*

---

### Description

Plots the log-likelihood at every iteration of the EM/CEM algorithm used to fit a MoEClust mixture model.

### Usage

```
MoE_plotLogLik(res,
               type = "l",
               xlab = "Iteration",
               ylab = "Log-Likelihood",
               xaxt = "n",
               ...)
```

### Arguments

| | |
|---|---|
| res | An object of class "MoEClust" generated by MoE_clust, or an object of class "MoECompare" generated by MoE_compare. Models with a noise component are facilitated here too. |
| type, xlab, ylab, xaxt | These graphical parameters retain their usual definitions from plot. |
| ... | Catches unused arguments, or additional arguments to be passed to plot. |

## Value

A plot of the log-likelihood versus the number EM iterations. A list with the vector of log-likelihood values and the final value at convergence can also be returned invisibly.

## Note

`plot.MoEClust` is a wrapper to `MoE_plotLogLik` which accepts the default arguments, and also produces other types of plots.

`res$LOGLIK` can also be plotted, to compare maximal log-likelihood values for all fitted models.

## Author(s)

Keefe Murphy - <<keefe.murphy@ucd.ie>>

## See Also

`MoE_clust`, `plot.MoEClust`,

## Examples

```
data(ais)
res <- MoE_clust(ais[,3:7], gating= ~ BMI, expert= ~ sex, tau0=0.1,
                 G=2, modelNames="EVE", network.data=ais)
(ll <- MoE_plotLogLik(res))
```

---

`MoE_stepwise` *Stepwise model/variable selection for MoEClust models*

---

## Description

Conducts a greedy forward stepwise search to identify the optimal `MoEClust` model according to some `criterion`. Components and/or `gating` covariates and/or `expert` covariates are added to new `MoE_clust` fits at each step, while each step is evaluated for all valid `modelNames`.

## Usage

```
MoE_stepwise(data,
             network.data = NULL,
             gating = NULL,
             expert = NULL,
             modelNames = NULL,
             noise = FALSE,
             criterion = c("bic", "icl", "aic"),
             equalPro = c("both", "yes", "no"),
             noise.gate = c("both", "yes", "no"),
             verbose = interactive(),
             ...)
```

## Arguments

data
: A numeric vector, matrix, or data frame of observations. Categorical variables are not allowed. If a matrix or data frame, rows correspond to observations and columns correspond to variables.

network.data
: An optional matrix or data frame in which to look for the covariates specified in the gating &/or expert networks, if any. Must include column names. Columns in network.data corresponding to columns in data will be automatically removed. While a single covariate can be supplied as a vector (provided the '$' operator is not used), it is safer to supply a named 1-column matrix or data frame in this instance.

gating
: A vector giving the names of columns in network.data used to define the scope of the gating network. The initial model will contain no covariates, thereafter all variables in gating will be considered for inclusion where appropriate.

  If gating is not supplied, *all* variables in network.data will be considered for the gating network. gating can also be supplied as NA, in which case *no* gating network covariates will ever be considered. Supplying gating and expert can be used to ensure different subsets of covariates enter different parts of the model.

expert
: A vector giving the names of columns in network.data used to define the scope of the expert network. The initial model will contain no covariates, thereafter all variables in expert will be considered for inclusion where appropriate.

  If expert is not supplied, *all* variables in network.data will be considered for the expert network. expert can also be supplied as NA, in which case *no* expert network covariates will ever be considered. Supplying expert and gating can be used to ensure different subsets of covariates enter different parts of the model.

modelNames
: A character string or valid model names, to be used to restrict the size of the search space, if desired. By default, *all* valid model types are explored. Rather than considered the changing of the model type as an additional step, every step is evaluated over all entries in modelNames. See MoE_clust for more details.

noise
: A logical indicating whether to assume all models contain an additional noise component (TRUE) or not (FALSE, the default). When TRUE, the search starts from a G=0 noise-only model, otherwise the search starts from a G=1 model with no covariates. See MoE_control for more details.

criterion
: The model selection criterion used to determine the optimal action at each step. Defaults to "bic".

equalPro
: A character string indicating whether models with equal mixing proportions should be considered. "both" (the default) means models with both equal and unequal mixing proportions will be considered, "yes" means only models with equal mixing proportions will be considered, and "no" means only models with unequal mixing proportions will be considered.

  Considering "both" equal and unequal mixing proportion models increases the search space and the computational burden, but this argument becomes irrelevant after a model, if any, with gating network covariate(s) is considered optimal for a given step. See MoE_control for more details.

noise.gate
: A character string indicating whether models where the gating network for the noise component depends on covariates are considered. "yes" means only models where this is the case will be considered, "no" means only models for which the noise component's mixing proportion is constant will be considered and "both" (the default) means both of these scenarios will be considered.

  Considering "both" increases the search space and the computational burden, but this argument is only relevant when noise=TRUE and gating covariates are being considered. See MoE_control for more details.

verbose
: Logical indicating whether to print messages pertaining to progress to the screen during fitting. By default is TRUE if the session is interactive, and FALSE otherwise. If FALSE, warnings and error messages will still be printed to the screen, but everything else will be suppressed.

...
: Additional arguments to MoE_control. Note that these arguments will be supplied to *all* candidate models for every step.

## Details

The arguments modelNames, equalPro, and noise.gate are provided for computational convenience. They can be used to reduce the number of models under consideration at each stage.

The same is true of the arguments gating and expert, which can each separately be made to consider all variables in network.data, or a subset, or none at all.

Without any prior information, it is best to accept the defaults at the expense of a longer run-time.

## Value

An object of class "MoECompare" containing information on all visited models and the optimal model (accessible via x$optimal).

## Note

It is advised to run this function once with noise=FALSE and once with noise=TRUE and then choose the optimal model across both sets of results.

At present, only additions (of components and covariates) are considered. In future updates, it will be possible to allow both additions and removals.

The function will attempt to remove duplicate variables found in both data and network.data; in particular, they will be removed from network.data. Users are however advised to careful specify data and network.data such that there are no duplicates, especially if the desired variable(s) should belong to network.data.

## Author(s)

Keefe Murphy - <<keefe.murphy@ucd.ie>>

## References

Murphy, K. and Murphy, T. B. (2019). Gaussian parsimonious clustering models with covariates and a noise component. *Advances in Data Analysis and Classification*, 1-33. <doi:10.1007/s11634-019-00373-8>.

### See Also

MoE_clust, MoE_compare, MoE_control

### Examples

```
# data(CO2data)
# Search over all models where the single covariate can enter either network
# (mod1 <- MoE_stepwise(CO2data$CO2, CO2data[,"GNP", drop=FALSE]))
#
# data(ais)
# Only look for EVE & EEE models with at most one expert network covariate
# Do not consider any gating covariates
# (mod2 <- MoE_stepwise(ais[,3:7], ais, gating=NA, expert="sex", modelNames=c("EVE", "EEE")))
#
# Look for models with a noise component, unequal mixing proportions,
# and only consider models with a constant mixing proportion for the noise component
# (mod3 <- MoE_stepwise(ais[,3:7], ais, noise=TRUE,  gating=c("SSF", "Ht"), expert="sex",
#                       equalPro="no", noise.gate="no", modelNames="EEE"))
#
# Compare both sets of results (with & without a noise component) for the ais data
# (comp <- MoE_compare(mod2, mod3, optimal.only=TRUE))
# comp$optimal
```

---

## MoE_Uncertainty *Plot Clustering Uncertainties*

---

### Description

Plots the clustering uncertainty for every observation from a fitted "MoEClust" model, including models with a noise component.

### Usage

```
MoE_Uncertainty(res,
                type = c("barplot", "profile"),
                truth = NULL,
                decreasing = FALSE,
                ...)
```

### Arguments

| | |
|---|---|
| res | An object of class "MoEClust" generated by MoE_clust, or an object of class "MoECompare" generated by MoE_compare. Models with a noise component are facilitated here too. |
| type | The type of plot to be produced (defaults to "barplot"). The "profile" option instead displays uncertainties in increasing/decreasing order of magnitude (see decreasing). |

`truth` An optional argument giving the true classification of the data. When `truth` is supplied and `type="barplot"`, misclassified observations are highlighted in a different colour, otherwise observations with uncertainty greater than `1/res$G` are given in a different colour. When `truth` is supplied and `type="profile"`, the uncertainty of misclassified observations is marked by vertical lines on the plot.

`decreasing` Logical indicating whether uncertainties should be ordered in decreasing order (defaults to `FALSE`). Only relevant when `type="profile"`.

`...` Catches unused arguments.

## Details

The y-axis of this plot runs from `0` to `1 -1/res$G`, with a horizontal line also drawn at `1/res$G`. When `type="barplot"`, uncertainties greater than this value are given a different colour when `truth` is not supplied, otherwise misclassified observations are given a different colour. Note, however, that $G^{(0)}$ = `res$G + 1` is used in place of `res$G` for models with a noise component.

## Value

A plot showing the clustering uncertainty of each observation (sorted in increasing/decreasing order when `type="profile"`). The (unsorted) vector of uncertainties can also be returned invisibly. When `truth` is supplied, the indices of the misclassified observations are also invisibly returned.

## Note

`plot.MoEClust` is a wrapper to `MoE_Uncertainty` which accepts the default arguments, and also produces other types of plots.

An alternative means of visualising clustering uncertainties (at least for multivariate data) is provided by the functions `MoE_gpairs` and `plot.MoEClust`, specifically when their argument `response.type` is given as `"uncertainty"`.

## Author(s)

Keefe Murphy - <<keefe.murphy@ucd.ie>>

## See Also

`MoE_clust`, `MoE_gpairs`, `plot.MoEClust`

## Examples

```
data(ais)
res <- MoE_clust(ais[,3:7], gating= ~ sex, G=3, modelNames="EEE", network.data=ais)

# Produce an uncertainty barplot
MoE_Uncertainty(res)

# Produce an uncertainty profile plot
MoE_Uncertainty(res, type="profile")
```

```
# Let's assume the true clusters correspond to sex
(ub <- MoE_Uncertainty(res, truth=ais$sex))
(up <- MoE_Uncertainty(res, type="profile", truth=ais$sex))
```

---

noise_vol *Approximate Hypervolume Estimate*

---

## Description

Computes simple approximations to the hypervolume of univariate and multivariate data sets. Also returns the location of the centre of mass.

## Usage

```
noise_vol(data,
          method = c("hypvol", "convexhull", "ellipsoidhull"),
          reciprocal = FALSE)
```

## Arguments

| | |
|---|---|
| data | A numeric vector, matrix, or data frame of observations. Categorical variables are not allowed, and covariates should not be included. If a matrix or data frame, rows correspond to observations and columns correspond to variables. There **must** be more observations than variables. |
| method | The method used to estimate the hypervolume. The default method uses the function hypvol. The "convexhull" and "ellipsoidhull" options require loading the geometry and cluster libraries, respectively. This argument is only relevant for multivariate data; for univariate data, the range of the data is used. |
| reciprocal | A logical variable indicating whether or not the reciprocal hypervolume is desired rather than the hypervolume itself. The default is to return the hypervolume. |

## Value

A list with the following two elements:

vol A hypervolume estimate (or its inverse).

This can be used as the hypervolume parameter for the noise component when observations are designated as noise in MoE_clust.

loc A vector of length ncol(data) giving the location of the centre of mass.

This can help in predicting the fitted values of models fitted with noise components via MoE_clust.

## Note

This function is called when adding a noise component to `MoEClust` models via the function `MoE_control`, specifically it's argument `noise.meth`. The function internally only uses the response variables, and not the covariates. However, one can bypass the invocation of this function by specifying its `noise.vol` argument directly. This is explicitly necessary for models for high-dimensional data which include a noise component for which this function cannot estimate a (hyper)volume.

Note that supplying the volume manually to `MoE_clust` can affect the summary of the means in `parameters$mean` and by extension the location of the MVN ellipses in `MoE_gpairs` plots for models with *both* expert network covariates and a noise component. The location cannot be estimated when the volume is supplied manually; in this case, prediction is made on the basis of renormalising the `z` matrix after discarding the column corresponding to the noise component. Otherwise, the mean of the noise component is accounted for. The renormalisation approach can be forced by specifying `noise.args$discard.noise=TRUE`, even when the mean of the noise component is available.

## Author(s)

Keefe Murphy - <<keefe.murphy@ucd.ie>>

## See Also

`hypvol`, `convhulln`, `ellipsoidhull`

## Examples

```
data(ais)
noise_vol(ais[,3:7], reciprocal=TRUE)

noise_vol(ais[,3:7], reciprocal=FALSE, method="convexhull")
```

---

# plot.MoEClust *Plot MoEClust Results*

---

## Description

Plot results for fitted MoE_clust mixture models with gating &/or expert network covariates: generalised pairs plots, model selection criteria, the log-likelihood vs. the EM iterations, and the gating network are all currently visualisable.

## Usage

```
## S3 method for class 'MoEClust'
plot(x,
     what = c("gpairs", "gating", "criterion", "loglik", "uncertainty"),
     ...)
```

## Arguments

- `x` — An object of class "MoEClust" generated by MoE_clust, or an object of class "MoECompare" generated by MoE_compare. Models with a noise component are facilitated here too.
- `what` — The type of graph requested:
  - `gpairs` A generalised pairs plot. To further customise this plot, arguments to MoE_gpairs can be supplied.
  - `gating` The gating network. To further customise this plot, arguments to MoE_plotGate and matplot can be supplied.
  - `criterion` The model selection criteria. To further customise this plot, arguments to MoE_plotCrit and plot.mclustBIC can be supplied.
  - `loglik` The log-likelihood vs. the iterations of the EM algorithm. To further customise this plot, arguments to MoE_plotLogLik and plot can be supplied.
  - `uncertainty` The clustering uncertainty for every observation. To further customise this plot, arguments to MoE_Uncertainty can be supplied.

  By default, all of the above graphs are produced.
- `...` — Optional arguments to be passed to MoE_gpairs, MoE_plotGate, MoE_plotCrit, MoE_plotLogLik, MoE_Uncertainty, matplot, plot.mclustBIC and plot. In particular, the argument legendArgs to plot.mclustBIC can be passed to MoE_plotCrit.

## Details

For more flexibility in plotting, use MoE_gpairs, MoE_plotGate, MoE_plotCrit, MoE_plotLogLik and MoE_Uncertainty directly.

## Value

The visualisation according to what of the results of a fitted MoEClust model.

## Note

Caution is advised producing generalised pairs plots when the dimension of the data is large.

Other types of plots are available by first calling as.Mclust on the fitted object, and then calling plot.Mclust on the results. These can be especially useful for univariate data.

## Author(s)

Keefe Murphy - <<keefe.murphy@ucd.ie>>

## References

Murphy, K. and Murphy, T. B. (2019). Gaussian parsimonious clustering models with covariates and a noise component. *Advances in Data Analysis and Classification*, 1-33. <doi:10.1007/s11634-019-00373-8>.

## See Also

MoE_clust, MoE_stepwise, MoE_gpairs, MoE_plotGate, MoE_plotCrit, MoE_plotLogLik, MoE_Uncertainty, as.Mclust, plot.Mclust

## Examples

```
data(ais)
res <- MoE_clust(ais[,3:7], gating= ~ BMI, expert= ~ sex,
                 G=2, modelNames="EVE", network.data=ais)

# Plot the gating network
plot(res, what="gating", x.axis=ais$BMI, xlab="BMI")

# Plot the log-likelihood
plot(res, what="loglik", col="blue")

# Plot the uncertainty profile
plot(res, what="uncertainty", type="profile")

# Produce a generalised pairs plot
plot(res, what="gpairs")

# Modify the gpairs plot by passing arguments to MoE_gpairs()
plot(res, what="gpairs", response.type="density",
     scatter.type="ci", jitter=FALSE, show.counts=FALSE)
```

---

# predict.MoEClust *Predictions for MoEClust models*

---

## Description

Predicts both cluster membership probabilities and fitted response values from a MoEClust model, using covariates and response data, or covariates only. The predicted MAP classification, mixing proportions, and component means are all also reported in both cases, as well as the predictions of the expert network corresponding to the most probable component.

## Usage

```
## S3 method for class 'MoEClust'
predict(object,
        newdata,
        resid = FALSE,
        discard.noise = FALSE,
        MAPresids = FALSE,
        use.y = TRUE,
        ...)

## S3 method for class 'MoEClust'
```

```
residuals(object,
          newdata,
          ...)
```

## Arguments

- `object` — An object of class "MoEClust" generated by MoE_clust, or an object of class "MoECompare" generated by MoE_compare. Predictions for models with a noise component are facilitated here too (see discard.noise).
- `newdata` — A list with two *named* components, each of which must be a data.frame or matrix with named columns, giving the data for which predictions are desired.
  - new.x The new covariates for the gating &/or expert networks. **Must** be supplied when newdata$new.y is supplied.
  - new.y (Optional) response data. When supplied, cluster and response prediction is based on both newdata$new.x and newdata$new.y, otherwise only on the covariates in newdata$new.x.

  If supplied as a list with elements new.x and new.y, both **must** have the same number of rows.

  Alternatively, a single data.frame or matrix can be supplied and an attempt will be made to extract & separate covariate and response columns (*if any*) into newdata$new.x and newdata$new.y based on the variable names in object$data and object$net.covs.

  When newdata is not supplied in any way, the covariates and response variables used in the fitting of the model are used here. It is possible to not supply new.y and to supply an empty data.frame or matrix for new.x (or to equivalently supply an empty data.frame or matrix for newdata itself) for models with no covariates of any kind, which effectively predicts the weighted mean of the component means.
- `resid` — A logical indicating whether to return the residuals also. Defaults to FALSE. Only allowed when response variables are supplied in some form. The function residuals is a wrapper to predict with the argument resid set to TRUE, with only the residuals returned.
- `discard.noise` — A logical governing how predictions of the responses are made for models with a noise component (otherwise this argument is irrelevant). By default (FALSE), the mean of the noise component is accounted for. Otherwise, or when the mean of the noise component is unavailable (due to having been manually supplied through MoE_control via noise.args$noise.vol), prediction of the responses is performed using a z matrix which is renormalised after discarding the column corresponding to the noise component. The renormalisation approach can be forced by specifying TRUE, even when the mean of the noise component is available. For models with a noise component fitted with algo="CEM", a small extra E-step is conducted for observations assigned to the non-noise components in this case.
- `MAPresids` — A logical indicating whether residuals are computed against y (TRUE, the default) or MAPy when FALSE. Not relevant for models with equal mixing proportions when only new.x is available. See **Value** below for more details.

| | |
|---|---|
| use.y | A logical indicating whether the response variables (supplied either via new.y or via newdata itself) are actually used in the prediction. Defaults to TRUE, but useful when FALSE for computing residuals as though only the covariates in new.x were supplied. |
| ... | Catches unused arguments (and allows the predict arguments discard.noise, MAPresids, and use.y to be passed through residuals). |

## Value

A list with the following named components, regardless of whether newdata$new.x and newdata$new.y were used, or newdata$new.x only.

| | |
|---|---|
| y | Aggregated fitted values of the response variables. |
| z | A matrix whose [i,k]-th entry is the probability that observation *i* of the newdata belongs to the *k*-th component. For models with a noise component, the final column gives the probability of belonging to the so-called *Cluster0*. |
| classification | The vector of predicted cluster labels for the newdata. 0 is returned for observations assigned to the noise component. |
| pro | The predicted mixing proportions for the newdata, i.e. predicted values of the gating network. object$parameters$pro is returned for models without gating network covariates. |
| mean | The predicted component means for the newdata, i.e. predicted values of the expert network. Given as a 3-dimensional array with dimensions given by the number of new observations, the number of variables, and the number of clusters. The first dimension is of length 1 when there are no expert network covariates, in which case the entries correspond to object$parameters$mean. |
| MAPy | Fitted values of the single expert network to which each observation is most probably assigned. Not returned for models with equal mixing proportions when only new.x is available. Likely to only be of use for models with gating and expert covariates when only new.x is supplied. Note that MAPy and y will coincide for models fitted via the CEM algorithm (see MoE_control and its argument algo). |

When residuals is called, only the residuals (governed by MAPresids) are returned; when predict is called with resid=TRUE, the list above will also contain the element resids, containing the residuals.

The returned values of pro and mean are always the same, regardless of whether newdata$new.x and newdata$new.y were used, or newdata$new.x only.

## Note

Predictions can also be made for models with a noise component, in which case z will include the probability of belonging to "Cluster0" & classification will include labels with the value 0 for observations classified as noise (if any). The argument discard.noise governs how the responses are predicted in the presence of a noise component (see noise_vol for more details).

Note that the argument discard.noise is invoked for any models with a noise component, while the similar MoE_control argument noise.args$discard.noise is only invoked for models with both a noise component and expert network covariates.

### Author(s)

Keefe Murphy - <<keefe.murphy@ucd.ie>>

### References

Murphy, K. and Murphy, T. B. (2019). Gaussian parsimonious clustering models with covariates and a noise component. *Advances in Data Analysis and Classification*, 1-33. <doi:10.1007/s11634-019-00373-8>.

### See Also

MoE_clust, MoE_control, noise_vol

### Examples

```
data(ais)

# Fit a MoEClust model and predict the same data
res     <- MoE_clust(ais[,3:7], G=2, gating=~BMI, expert=~sex,
                     modelNames="EVE", network.data=ais)
pred1   <- predict(res)

# Remove some rows of the data for prediction purposes
ind     <- sample(1:nrow(ais), 5)
dat     <- ais[-ind,]

# Fit another MoEClust model to the retained data
res2    <- MoE_clust(dat[,3:7], G=3, gating=~BMI + sex,
                     modelNames="EEE", network.data=dat)

# Predict held back data using the covariates & response variables
(pred2  <- predict(res2, newdata=ais[ind,]))
# pred2 <- predict(res2, newdata=list(new.y=ais[ind,3:7],
#                                     new.x=ais[ind,c("BMI", "sex")]))

# Get the residuals
residuals(res2, newdata=ais[ind,])

# Predict held back data using only the covariates
(pred3  <- predict(res2, newdata=list(new.x=ais[ind,c("BMI", "sex")])))
# pred3 <- predict(res2, newdata=ais[ind,c("BMI", "sex")])
```

---

## quant_clust — *Quantile-Based Clustering for Univariate Data*

---

### Description

Returns a quantile-based clustering for univariate data.

## Usage

```
quant_clust(x,
            G)
```

## Arguments

| | |
|---|---|
| `x` | A vector of numeric data. |
| `G` | The desired number of clusters. |

## Value

The vector of cluster labels.

## Examples

```
data(CO2data)
quant_clust(CO2data$CO2, G=2)
```

# Index


∗Topic **clustering**
MoE_clust, 14
MoE_compare, 21
MoE_crit, 32
MoE_cstep, 34
MoE_dens, 36
MoE_estep, 37
MoE_stepwise, 51
predict.MoEClust, 59
quant_clust, 62
∗Topic **control**
aitken, 6
MoE_control, 25
noise_vol, 56
∗Topic **datasets**
ais, 5
CO2data, 9
∗Topic **main**
MoE_clust, 14
MoE_compare, 21
MoE_stepwise, 51
plot.MoEClust, 57
predict.MoEClust, 59
∗Topic **package**
MoEClust-package, 2
∗Topic **plotting**
MoE_gpairs, 39
MoE_plotCrit, 48
MoE_plotGate, 49
MoE_plotLogLik, 50
MoE_Uncertainty, 54
plot.MoEClust, 57
∗Topic **utility**
as.Mclust, 7
drop_constants, 10
drop_levels, 11
expert_covar, 12
force_posiDiag, 14
MoE_mahala, 45
MoE_news, 47
predict.MoEClust, 59

ais, 5
aitken, *3*, 6, *18*, *20*, *28*, *31*
as.Mclust, *3*, 7, *8*, *13*, *20*, *24*, *58*, *59*

clustCombi, *8*
clustCombiOptim, *8*
clustMD, *26–31*
CO2data, 9
convhulln, *31*, *57*

drop_constants, 10, *12*, *28*
drop_levels, *11*, 11, *28*

ellipsoidhull, *31*, *57*
expert_covar, *8*, *9*, 12, *19*, *20*, *41*, *42*, *44*

force_posiDiag, 14
formula, *10*, *16*, *17*

hc, *25*, *26*, *29*, *31*
hypvol, *26*, *31*, *56*, *57*

I, *10*, *11*, *16*, *17*, *20*

lm, *11*, *12*, *45*
logLik.Mclust, *8*

matplot, *49*, *50*, *58*
Mclust, *7*, *9*, *25*, *26*, *29*
mclust.options, *31*, *49*
mclustModelNames, *15*, *20*, *23*, *24*, *32*, *33*
mclustVariance, *19*, *20*, *34–39*
MoE_clust, *3*, *6–9*, *13*, 14, *16*, *17*, *21*, *22*, *24–27*, *29–31*, *33–40*, *44*, *48–52*, *54–60*, *62*
MoE_compare, *3*, *8*, *13*, *17*, *20*, 21, *22*, *23*, *30*, *31*, *40*, *48–50*, *54*, *58*, *60*
MoE_control, *3*, *6*, *7*, *14–20*, *23*, 25, *30*, *35*, *39*, *52–54*, *60–62*

MoE_crit, *3*, *18*, *20*, 32
MoE_cstep, *20*, 34, *36, 37*, *39*
MoE_dens, *3*, *20*, *34, 35*, 36, *37–39*
MoE_estep, *3*, *20*, *35–37*, 37
MoE_gpairs, *3*, *8*, *13*, *26, 27*, 39, *44*, *55*, *57–59*
MoE_mahala, 45
MoE_news, 47
MoE_plotCrit, *48*, 48, *58, 59*
MoE_plotGate, *49*, 49, *58, 59*
MoE_plotLogLik, 50, *51*, *58, 59*
MoE_stepwise, *3*, *20–22*, *24*, *30, 31*, *44*, 51, *59*
MoE_Uncertainty, *40*, *44*, 54, *55*, *58, 59*
MoEClust *(*MoEClust-package*)*, 2
MoEClust-package, 2
multinom, *18*, *30, 31*

noise_vol, *26, 27*, *31*, *35, 36*, *38*, 56, *61, 62*
nVarParams, *33*

panel.bwplot, *41*, *44*
panel.stripplot, *41*, *44*
panel.violin, *44*
plot, *8*, *20*, *50*, *58*
plot.Mclust, *8, 9*, *58, 59*
plot.mclustBIC, *48*, *58*
plot.MoEClust, *3*, *9*, *13*, *20*, *24*, *44*, *48–51*, *55*, 57
predict, *3*, *20*
predict.MoEClust, *20*, 59
print.MoEClust *(*MoE_clust*)*, 14
print.MoECompare *(*MoE_compare*)*, 21

quant_clust, *25*, *31*, 62

residuals.MoEClust *(*predict.MoEClust*)*, 59
rowLogSumExps, *38, 39*

strucplot, *41*, *43, 44*
summary.MoEClust *(*MoE_clust*)*, 14