Package ‘flexmix’

February 19, 2015

Type Package
Title Flexible Mixture Modeling
Version 2.3-13
Description FlexMix implements a general framework for finite
mixtures of regression models using the EM algorithm.
FlexMix provides the E-step and all data handling, while the M-step can be
supplied by the user to easily define new models. Existing drivers
implement mixtures of standard linear models, generalized linear
models and model-based clustering.
Depends R (>= 2.15.0), lattice
Imports grid, methods, modeltools (>= 0.2-16), nnet, stats, stats4
Suggests codetools, diptest, ellipse, gclus, glmnet, lme4 (>= 1.0),
MASS, mgcv (>= 1.8-0), mlbench, mlogit, multcomp, mvtnorm,
survival
License GPL (>= 2)
LazyLoad yes
Author Bettina Gruen [aut, cre],
  Friedrich Leisch [aut],
  Deepayan Sarkar [ctb],
  Frederic Mortier [ctb],
  Nicolas Picard [ctb]
Maintainer Bettina Gruen <Bettina.Gruen@jku.at>
NeedsCompilation no
Repository CRAN
Date/Publication 2015-01-17 18:51:41

R topics documented:

AIC-methods ................................................................. 3
betablocker ................................................................. 3
BIC-methods ................................................................. 4
bioChemists ................................................................. 4
boot ................................................................. 5
BregFix ............................................................... 7
candy ................................................................. 7
dmft ................................................................. 8
EIC ................................................................. 9
ExLinear ............................................................ 10
ExNclus ............................................................. 12
ExNPreg ............................................................ 12
fabricfault .......................................................... 13
fitted-methods ...................................................... 14
flexmix .............................................................. 14
flexmix-class ....................................................... 17
FLXcomponent-class ............................................. 18
FLXcontrol-class ................................................ 19
FLXdist ............................................................ 20
FLXdist-class ...................................................... 21
FLXfit ............................................................. 22
FLXM-class ........................................................ 23
FLXMCfactanal .................................................... 24
FLXMCmvbinary ................................................... 25
FLXCMvcombi ..................................................... 26
FLXCMvnorm ....................................................... 27
FLXCMvpois ......................................................... 29
FLXMRcondlogit ................................................... 30
FLXMRglm ........................................................ 31
FLXMRglmfix ....................................................... 32
FLXMRglmnet ...................................................... 33
FLXMRlmer ........................................................ 35
FLXMRlmmc ........................................................ 36
FLXMRmgcv ......................................................... 37
FLXMRmultinom ................................................... 39
FLXMRrobglm ...................................................... 40
FLXMRziglm ....................................................... 41
FLXnested-class .................................................. 42
FLXP ............................................................... 43
FLXP-class ........................................................ 43
group ............................................................. 44
ICL ............................................................... 45
KLdiv ............................................................. 46
Lapply-methods ................................................... 47
logLik-methods ................................................... 48
Mehta ............................................................. 49
NregFix ........................................................... 50
patent ............................................................. 50
plot-methods ...................................................... 51
plotEll .......................................................... 53
posterior ......................................................... 54
AIC-methods

refit-methods ............................................................. 54
relabel ................................................................. 57
flexmix .............................................................. 58
salmonellaTA98 ......................................................... 59
seizure .............................................................. 60
stepFlexmix .......................................................... 61
tribolium .......................................................... 64
trypanosome .......................................................... 65
whiskey ............................................................. 66

Index 67

<table>
<thead>
<tr>
<th>AIC-methods</th>
<th>Methods for Function AIC</th>
</tr>
</thead>
</table>

Description

Compute the Akaike Information Criterion.

Methods

object = flexmix: Compute the AIC of a flexmix object

object = stepFlexmix: Compute the AIC of all models contained in the stepFlexmix object.

betablocker         Clinical Trial of Beta-Blockers

Description

22-centre clinical trial of beta-blockers for reducing mortality after myocardial infarction.

Usage

data("betablocker")

Format

A data frame with 44 observations on the following 4 variables.

Deaths  Number of deaths.
Total   Total number of patients.
Center  Number of clinical centre.
Treatment  A factor with levels Control and Treated.
Source


References


Examples

data("betablocker", package = "flexmix")
betaMix <- initFlexmix(cbind(Deaths, Total - Deaths) ~ 1 | Center,
data = betablocker, k = 3, nrep = 5,
model = FLXMRglmfix(family = "binomial",
fixed = ~ Treatment))

---

BIC-methods

Methods for Function BIC

Description

Compute the Bayesian Information Criterion.

Methods

object = flexmix: Compute the BIC of a flexmix object

object = stepFlexmix: Compute the BIC of all models contained in the stepFlexmix object.

---

bioChemists

Articles by Graduate Students in Biochemistry Ph.D. Programs

Description

A sample of 915 biochemistry graduate students.

Usage

data("bioChemists")
Format

- **art** count of articles produced during last 3 years of Ph.D.
- **fem** factor indicating gender of student, with levels Men and Women
- **mar** factor indicating marital status of student, with levels Single and Married
- **kid5** number of children aged 5 or younger
- **phd** prestige of Ph.D. department
- **ment** count of articles produced by Ph.D. mentor during last 3 years

Details

This data set is taken from package **pscl** provided by Simon Jackman.

Source

found in Stata format at [http://www.indiana.edu/~jslsoc/stata/socdata/couart2.dta](http://www.indiana.edu/~jslsoc/stata/socdata/couart2.dta)

References


---

**boot**

*Bootstrap a flexmix Object*

**Description**

Given a `flexmix` object perform parametric or empirical bootstrap.

**Usage**

```r
boot(object, ...)  # S4 method for signature 'flexmix'
boot(object, R, sim = c("ordinary", "empirical", "parametric"),
      initialize_solution = FALSE, keep_weights = FALSE,
      keep_groups = TRUE, verbose = 0, control,
      k, model = FALSE, ...)  
LR_test(object, ...)  # S4 method for signature 'flexmix'
LR_test(object, R, alternative = c("greater", "less"), control, ...)
```
Arguments

object  A fitted finite mixture model of class \texttt{flexmix}.

\text{R}  The number of bootstrap replicates.

\text{sim}  A character string indicating the type of simulation required. Possible values are "ordinary" (the default), "parametric", or "empirical".

\text{initialize\_solution}  A logical. If \text{TRUE} the EM algorithm is initialized in the given solution.

\text{keep\_weights}  A logical. If \text{TRUE} the weights are kept.

\text{keep\_groups}  A logical. If \text{TRUE} the groups are kept.

\text{verbose}  If a positive integer, then progress information is reported every \text{verbose} iterations. If 0, no output is generated during the bootstrap replications.

\text{control}  Object of class \texttt{FLXcontrol} or a named list. If missing the control of the fitted object is taken.

\text{k}  Vector of integers specifying for which number of components finite mixtures are fitted to the bootstrap samples. If missing the number of components of the fitted object are taken.

\text{alternative}  A character string specifying the alternative hypothesis, must be either "greater" (default) or "less" indicating if the alternative hypothesis is that the mixture has one more component or one less.

\text{model}  A logical. If \text{TRUE} the model and the weights slot for each sample are stored and returned.

\text{...}  Further arguments to be passed to or from methods.

Value

\text{boot} returns an object of class \texttt{FLXboot} which contains the fitted parameters, the fitted priors, the log likelihoods, the number of components of the fitted mixtures and the information if the EM algorithm has converged.

\text{LR\_test} returns an object of class \texttt{htest} containing the number of valid bootstrap replicates, the p-value, the - twice log likelihood ratio test statistics for the original data and the bootstrap replicates.

Author(s)

Bettina Gruen

Examples

data("NPreg", package = "flexmix")
fitted <- initFlexmix(yn ~ x + I(x^2) | id2, data = NPreg, k = 2)
## Not run:
lrtest <- LR_test(fitted, alternative = "greater", R = 20, verbose = 1)
## End(Not run)
Description

A simple artificial regression example data set with 3 latent classes, one independent variable $x$ and a concomitant variable $w$.

Usage

data("BregFix")

Format

A data frame with 200 observations on the following 5 variables.

- yes: number of successes
- no: number of failures
- x: independent variable
- w: concomitant variable, a factor with levels 0 1
- class: latent class memberships

Examples

data("BregFix", package = "flexmix")
Model <- FLXMRglmfix(family="binomial",
  nested = list(formula = c(~x, ~0), k = c(2, 1)))
Conc <- FLXPtrivnunif(-w)
FittedBin <- initFlexmix(cbind(yes, no) ~ 1, data = BregFix,
  k = 3, model = Model, concomitant = Conc)
summary(FittedBin)

candy  Candy Packs Purchased

Description

The data is from a new product and concept test where the number of individual packs of hard candy purchased within the past 7 days is recorded.

Usage

data("candy")
**dmft**

**Dental Data**

**Format**

A data frame with 21 observations on the following 2 variables.

- **Packages** a numeric vector
- **Freq** a numeric vector

**Source**


**References**


**Description**

Count data from a dental epidemiological study for evaluation of various programs for reducing caries collected among school children from an urban area of Belo Horizonte (Brazil).

**Usage**

data("dmft")

**Format**

A data frame with 797 observations on the following 5 variables.

- **End** Number of decayed, missing or filled teeth at the end of the study.
- **Begin** Number of decayed, missing or filled teeth at the beginning of the study.
- **Gender** A factor with levels male and female.
- **Ethnic** A factor with levels brown, white and black.
- **Treatment** A factor with levels control, educ, enrich, rinse, hygiene and all.
Details

The aim of the caries prevention study was to compare four methods to prevent dental caries. Interventions were carried out according to the following scheme:

- **control** Control group
- **educ** Oral health education
- **enrich** Enrichment of the school diet with rice bran
- **rinse** Mouthwash with 0.2% sodium fluoride (NaF) solution
- **hygiene** Oral hygiene
- **all** All four methods together

Source


Examples

```r
data("dmft", package = "flexmix")
dmft_flx <- initFlexmix(End ~ 1, data = dmft, k = 2,
model = FLXMRglmfix(family = "poisson",
fixed = ~ Gender + Ethnic + Treatment))
```

---

### EIC

**Entropic Measure Information Criterion**

**Description**

Compute the Entropic measure Information criterion for model selection.

**Usage**

```r
## S4 method for signature 'flexmix'
EIC(object, ...)
## S4 method for signature 'stepFlexmix'
EIC(object, ...)
```

**Arguments**

- **object** see Methods section below
- **...** Some methods for this generic function may take additional, optional arguments. At present none do.

**Value**

Returns a numeric vector with the corresponding EIC value(s).
Methods

object = "flexmix": Compute the EIC of a flexmix object.

object = "stepFlexmix": Compute the EIC of all models contained in the stepFlexmix object.

Author(s)

Bettina Gruen

References


Examples

data("NPreg", package = "flexmix")
ex1 <- flexmix(y ~ x + I(x^2), data = NPreg, k = 2)
EIC(ex1)

---

ExLinear  
*Artificial Data from a Generalized Linear Regression Mixture*

Description

Generate random data mixed from k generalized linear regressions (GLMs).

Usage

ExLinear(beta, n, xdist = "runif", xdist.args = NULL,
family = c("gaussian","poisson"), sd = 1, ...)

Arguments

- **beta**
  - A matrix of regression coefficients. Each row corresponds to a variable, each column to a mixture component. The first row is used as intercept.

- **n**
  - Integer, the number of observations per component.

- **xdist**
  - Character, name of a random number function for the explanatory variables.

- **xdist.args**
  - List, arguments for the random number functions.

- **family**
  - A character string naming a GLM family. Only "gaussian" and "poisson" are implemented at the moment.

- **sd**
  - Numeric, the error standard deviation for each component for Gaussian responses.

- **...**
  - Used as default for xdist.args if that is not specified.
Details

First, arguments \( n \) (and \( sd \) for Gaussian response) are recycled to the number of mixture components \( ncol(beta) \), and arguments \( xdist \) and \( xdist.args \) are recycled to the number of explanatory variables \( nrow(beta) \). Then a design matrix is created for each mixture component by drawing random numbers from \( xdist \). For each component, the design matrix is multiplied by the regression coefficients to form the linear predictor. For Gaussian responses the identity link is used, for Poisson responses the log link.

The true cluster memberships are returned as attribute "clusters".

Examples

```r
## simple example in 2d
beta <- matrix(c(1, 2, 3, -1), ncol = 2)
sigma <- c(.5, 1)
df1 <- ExLinear(beta, 100, sd = sigma, min = -1, max = 2)
plot(y ~ x1, df1, col = attr(df1, "clusters"))

## add a second explanatory variable with exponential distribution
beta2 <- rbind(beta, c(-2, 2))
df2 <- ExLinear(beta2, 100, sd = c(.5, 1),
                xdist = c("runif", "rexp"),
                xdist.args = list(list(min = -1, max = 2),
                              list(rate = 3)))
summary(df2)

opar = par("mfrow")
par(mfrow = 1:2)
hist(df2$x1)
hist(df2$x2)
par(opar)

f1 <- flexmix(y ~ ., data = df2, k = 2)

## sort components on Intercept
f1 <- relabel(f1, "model", "Intercept")

## the parameters should be close to the true beta and sigma
round(parameters(f1), 3)
rbind(beta2, sigma)

### A simple Poisson GLM
df3 <- ExLinear(beta/2, 100, min = -1, max = 2, family = "poisson")
plot(y ~ x1, df3, col = attr(df3, "clusters"))

f3 <- flexmix(y ~ ., data = df3, k = 2,
              model = FLXMRglm(family = "poisson"))
round(parameters(relabel(f3, "model", "Intercept")), 3)
beta/2
```
ExNclus

**Artificial Example with 4 Gaussians**

**Description**
A simple artificial example for normal clustering with 4 latent classes, all of them having a Gaussian distribution. See the function definition for true means and covariances.

**Usage**
```
exnclus(n)
data("Nclus")
```

**Arguments**
n Number of observations in the two small latent classes.

**Details**
The Nclus data set can be re-created by ExNclus(100) using set.seed(2602), it has been saved as a data set for simplicity of examples only.

**Examples**
```
data("Nclus", package = "flexmix")
require("MASS")
eqscplot(Nclus, col = rep(1:4, c(100, 100, 150, 200)))
```

---

ExNPreg

**Artificial Example for Normal, Poisson and Binomial Regression**

**Description**
A simple artificial regression example with 2 latent classes, one independent variable (uniform on [0, 10]), and three dependent variables with Gaussian, Poisson and Binomial distribution, respectively.

**Usage**
```
exnpreg(n)
data("NPreg")
```

**Arguments**
n Number of observations per latent class.
**Details**

The `npreg` data frame can be re-created by `ExNPrege(100)` using `set.seed(2602)`, it has been saved as a data set for simplicity of examples only.

**Examples**

```r
data("NPreg", package = "flexmix")
plot(yn ~ x, data = NPreg, col = class)
plot(yp ~ x, data = NPreg, col = class)
plot(yb ~ x, data = NPreg, col = class)
```

---

**Fabric Faults**

**Description**

Number of faults in rolls of a textile fabric.

**Usage**

```r
data("fabricfault")
```

**Format**

A data frame with 32 observations on the following 2 variables.

- **Length** Length of role (m).
- **Faults** Number of faults.

**Source**


**References**


**Examples**

```r
data("fabricfault", package = "flexmix")
fabricMix <- initFlexmix(Faults ~ 1, data = fabricfault, k = 2,
model = FLXMRglmfix(family = "poisson",
fixed = ~ log(Length)),
nrep = 5)
```
fitted-methods

 Extract Model Fitted Values

Description

Extract fitted values for each component from a flexmix object.

Usage

```r
## S4 method for signature 'flexmix'
fitted(object, drop = TRUE, aggregate = FALSE, ...)
```

Arguments

- `object`: an object of class "flexmix" or "FLXR"
- `drop`: logical, if TRUE then the function tries to simplify the return object by combining lists of length 1 into matrices.
- `aggregate`: logical, if TRUE then the fitted values for each model aggregated over the components are returned.
- `...`: currently not used

Author(s)

Friedrich Leisch and Bettina Gruen

Examples

```r
data("NPreg", package = "flexmix")
ex1 <- flexmix(yn ~ x + I(x^2), data = NPreg, k = 2)
matplot(NPreg$x, fitted(ex1), pch = 16, type = "p")
points(NPreg$x, NPreg$yn)
```

---

flexmix

Flexible Mixture Modeling

Description

FlexMix implements a general framework for finite mixtures of regression models. Parameter estimation is performed using the EM algorithm: the E-step is implemented by flexmix, while the user can specify the M-step.
Usage

```r
flexmix(formula, data = list(), k = NULL, cluster = NULL, 
         model = NULL, concomitant = NULL, control = NULL, 
         weights = NULL)
```

## S4 method for signature 'flexmix'

```r
summary(object, eps = 1e-4, ...)
```

Arguments

- `formula`: A symbolic description of the model to be fit. The general form is `y ~ x | g` where `y` is the response, `x` the set of predictors and `g` an optional grouping factor for repeated measurements.
- `data`: An optional data frame containing the variables in the model.
- `k`: Number of clusters (not needed if `cluster` is specified).
- `cluster`: Either a matrix with `k` columns of initial cluster membership probabilities for each observation; or a factor or integer vector with the initial cluster assignments of observations at the start of the EM algorithm. Default is random assignment into `k` clusters.
- `weights`: An optional vector of replication weights to be used in the fitting process. Should be `NULL`, an integer vector or a formula.
- `model`: Object of class `FLXm` or list of `FLXm` objects. Default is the object returned by calling `FLXmRglm()`.
- `concomitant`: Object of class `FLXp`. Default is the object returned by calling `FLXpconstant`.
- `control`: Object of class `FLXcontrol` or a named list.
- `object`: Object of class `flexmix`.
- `eps`: Probabilities below this threshold are treated as zero in the summary method.
- `...`: Currently not used.

Details

FlexMix models are described by objects of class `FLXm`, which in turn are created by driver functions like `FLXmRglm` or `FLXmCmvnorm`. Multivariate responses with independent components can be specified using a list of `FLXm` objects.

The `summary` method lists for each component the prior probability, the number of observations assigned to the corresponding cluster, the number of observations with a posterior probability larger than `eps` and the ratio of the latter two numbers (which indicates how separated the cluster is from the others).

Value

Returns an object of class `flexmix`.

Author(s)

Friedrich Leisch and Bettina Gruen
References


See Also

plot-methods

Examples

data("NPreg", package = "flexmix")

## mixture of two linear regression models. Note that control parameters
## can be specified as named list and abbreviated if unique.
ex1 <- flexmix(yn ~ x + I(x^2), data = NPreg, k = 2,
               control = list(verb = 5, iter = 100))

ex1
summary(ex1)
plot(ex1)

## now we fit a model with one Gaussian response and one Poisson
## response. Note that the formulas inside the call to FLXMRglm are
## relative to the overall model formula.
ex2 <- flexmix(yn ~ x, data = NPreg, k = 2,
               model = list(FLXMRglm(yn ~ . + I(x^2)),
                             FLXMRglm(yp ~ ., family = "poisson")))

plot(ex2)

ex2
table(ex2@cluster, NPreg$class)

## for Gaussian responses we get coefficients and standard deviation
parameters(ex2, component = 1, model = 1)

## for Poisson response we get only coefficients
parameters(ex2, component = 1, model = 2)

## fitting a model only to the Poisson response is of course
## done like this
ex3 <- flexmix(yp ~ x, data = NPreg, k = 2,
               model = FLXMRglm(family = "poisson"))

## if observations are grouped, i.e., we have several observations per
## individual, fitting is usually much faster:
flexmix-class

```r
ex4 <- flexmix(y~x|id1, data = NPreg, k = 2,
    model = FLXMRglm(family = "poisson"))

## And now a binomial example. Mixtures of binomials are not generically
## identified, here the grouping variable is necessary:
set.seed(1234)
ex5 <- initFlexmix(cbind(yb, 1 - yb) ~ x, data = NPreg, k = 2,
    model = FLXMRglm(family = "binomial"), nrep = 5)
table(NPreg$class, clusters(ex5))

ex6 <- initFlexmix(cbind(yb, 1 - yb) ~ x | id2, data = NPreg, k = 2,
    model = FLXMRglm(family = "binomial"), nrep = 5)
table(NPreg$class, clusters(ex6))
```

---

**flexmix-class**

Class "flexmix"

---

**Description**

A fitted `flexmix` model.

**Slots**

- `model`: List of `FLX` objects.
- `prior`: Numeric vector with prior probabilities of clusters.
- `posterior`: Named list with elements scaled and unscaled, both matrices with one row per observation and one column per cluster.
- `iter`: Number of EM iterations.
- `k`: Number of clusters after EM.
- `k0`: Number of clusters at start of EM.
- `cluster`: Cluster assignments of observations.
- `size`: Cluster sizes.
- `logLik`: Log-likelihood at EM convergence.
- `df`: Total number of parameters of the model.
- `components`: List describing the fitted components using `FLXcomponent` objects.
- `formula`: Object of class "formula".
- `control`: Object of class "FLXcontrol".
- `call`: The function call used to create the object.
- `group`: Object of class "factor".
- `converged`: Logical, `TRUE` if EM algorithm converged.
- `concomitant`: Object of class "FLXP".
- `weights`: Optional weights of the observations.
Extends
Class FLXdist, directly.

Accessor Functions
The following functions should be used for accessing the corresponding slots:

cluster: Cluster assignments of observations.
posterior: A matrix of posterior probabilities for each observation.

Author(s)
Friedrich Leisch and Bettina Gruen

FLXcomponent-class  Class "FLXcomponent"

Description
A fitted component of a flexmix model.

Objects from the Class
Objects can be created by calls of the form new("FLXcomponent", ...).

Slots
df: Number of parameters used by the component.
logLik: Function computing the log-likelihood of observations.
parameters: List with model parameters.
predict: Function predicting response for new data.

Author(s)
Friedrich Leisch and Bettina Gruen
Description

Hyperparameters for the EM algorithm.

Objects from the Class

Objects can be created by calls of the form `new("FLXcontrol", ...`). In addition, named lists can be coerced to `FLXcontrol` objects, names are completed if unique (see examples).

Slots

- `iter.max`: Maximum number of iterations.
- `minprior`: Minimum prior probability of clusters, components falling below this threshold are removed during the iteration.
- `tolerance`: The EM algorithm is stopped when the (relative) change of log-likelihood is smaller than `tolerance`.
- `verbose`: If a positive integer, then the log-likelihood is reported every `verbose` iterations. If 0, no output is generated during model fitting.
- `classify`: Character string, one of "auto", "weighted", "hard" (or "CEM"), "random" or "SEM".
- `nrep`: Reports the number of random initializations used in `stepFlexmix()` to determine the mixture.

Run `new("FLXcontrol")` to see the default settings of all slots.

Author(s)

Friedrich Leisch and Bettina Gruen

Examples

```r
## have a look at the defaults
new("FLXcontrol")

## corce a list
mycont <- list(iter = 200, tol = 0.001, class = "r")
as(mycont, "FLXcontrol")
```
FLXdist

Finite Mixtures of Distributions

Description

Constructs objects of class FLXdist which represent unfitted finite mixture models.

Usage

   FLXdist(formula, k = NULL, model = FLXMRglm(), components,
            concomitant = FLXPconstant())

Arguments

   formula       A symbolic description of the model to be fit. The general form is \(~x | g\) where \(x\) is the set of predictors and \(g\) an optional grouping factor for repeated measurements.

   k             Integer specifying the number of cluster or a numeric vector of length equal to the length of components, specifying the prior probabilities of clusters.

   model        Object of class FLXM or a list of FLXM objects. Default is the object returned by calling FLXMRglm().

   components   A list of length equal to the number of components containing a list of length equal to the number of models which again contains a list of named elements for defining the parameters of the component-specific model.

   concomitant  Object of class FLXconcomitant specifying the model for concomitant variables.

Value

   Returns an object of class FLXdist.

Author(s)

   Bettina Gruen

See Also

   FLXdist-class
FLXdist-class

Class "FLXdist"

Description

Objects of class FLXdist represent unfitted finite mixture models.

Usage

```r
## S4 method for signature 'FLXdist'
parameters(object, component = NULL, model = NULL, which = c("model",
            "concomitant"), simplify = TRUE, drop = TRUE)
## S4 method for signature 'FLXdist'
predict(object, newdata = list(), aggregate = FALSE, ...)
```

Arguments

- **object**: An object of class "FLXdist".
- **component**: Number of component(s), if NULL all components are returned.
- **model**: Number of model(s), if NULL all models are returned.
- **which**: Specifies if the parameters of the component specific model or the concomitant variable model are returned.
- **simplify**: Logical, if TRUE the returned values are simplified to a vector or matrix if possible.
- **drop**: Logical, if TRUE the function tries to simplify the return object by omitting lists of length one.
- **newdata**: Dataframe containing new data.
- **aggregate**: Logical, if TRUE then the predicted values for each model aggregated over the components are returned.
- **...**: Passed to the method of the model class.

Slots

- **model**: List of FLXM objects.
- **prior**: Numeric vector with prior probabilities of clusters.
- **components**: List describing the components using FLXcomponent objects.
- **concomitant**: Object of class "FLXP".
- **formula**: Object of class "formula".
- **call**: The function call used to create the object.
- **k**: Number of clusters.
Accessor Functions

The following functions should be used for accessing the corresponding slots:

- parameters: The parameters for each model and component, return value depends on the model.
- prior: Numeric vector of prior class probabilities/component weights

Author(s)

Friedrich Leisch and Bettina Gruen

See Also

FLXdist

Description

This is the basic computing engine called by flexmix, it should usually not be used directly.

Usage

\[
\text{FLXfit(model, concomitant, control, postunscaled = NULL, groups, weights)}
\]

Arguments

- model: List of FLX objects.
- concomitant: Object of class FLXP.
- control: Object of class FLXcontrol.
- weights: A numeric vector of weights to be used in the fitting process.
- postunscaled: Initial a-posteriori probabilities of the observations at the start of the EM algorithm.
- groups: List with components group which is a factor with optional grouping of observations and groupfirst which is a logical vector for the first observation of each group.

Value

Returns an object of class flexmix.

Author(s)

Friedrich Leisch and Bettina Gruen
**FLXM-class**

**See Also**

`flexmix`, `flexmix-class`

---

**FLXM-class**  
Class “FLXM”

---

**Description**

FlexMix model specification.

**Objects from the Class**

Objects can be created by calls of the form `new("FLXM", ...)`, typically inside driver functions like `FLXMRglm` or `FLXCMvnorm`.

**Slots**

- `fit`: Function returning an FLXcomponent object.
- `defineComponent`: Expression to determine the FLXcomponent object given the parameters.
- `weighted`: Logical indicating whether `fit` can do weighted likelihood maximization.
- `name`: Character string used in print methods.
- `formula`: Formula describing the model.
- `fullformula`: Resulting formula from updating the model formula with the formula specified in the call to `flexmix`.
- `x`: Model matrix.
- `y`: Model response.
- `terms`, `xlevels`, `contrasts`: Additional information for model matrix.
- `preproc.x`: Function for preprocessing matrix `x` before the EM algorithm starts, by default the identity function.
- `preproc.y`: Function for preprocessing matrix `y` before the EM algorithm starts, by default the identity function.

**Author(s)**

Friedrich Leisch and Bettina Gruen
FLXMCfactanal  

*Driver for Mixtures of Factor Analyzers*

**Description**

This driver for `flexmix` implements estimation of mixtures of factor analyzers using ML estimation of factor analysis implemented in `factanal` in each M-step.

**Usage**

`FLXMCfactanal(formula = . ~ ., factors = 1, ...)`

**Arguments**

- `formula` A formula which is interpreted relative to the formula specified in the call to `flexmix` using `update.formula`. Only the left-hand side (response) of the formula is used. Default is to use the original `flexmix` model formula.
- `factors` Integer specifying the number of factors in each component.
- `...` Passed to `factanal`

**Value**

`FLXMCfactanal` returns an object of class FLXM.

**Warning**

This does not implement the AECM framework presented in McLachlan and Peel (2000, p.245), but uses the available functionality in R for ML estimation of factor analyzers. The implementation therefore is only experimental and has not been well tested.

Please note that in general a good initialization is crucial for the EM algorithm to converge to a suitable solution for this model class.

**Author(s)**

Bettina Gruen

**References**


**See Also**

`flexmix`
Examples

```r
## Reproduce (partly) Table 8.1. p.255 (McLachlan and Peel, 2000)
if (require("gclus")) {
  data("wine", package = "gclus")
  wine_data <- as.matrix(wine[,,-1])
  set.seed(123)
  wine_fl_diag <- initFlexmix(wine_data ~ 1, k = 3, nrep = 10,
    model = FLXMCmvnorm(diagonal = TRUE))
  wine_fl_fact <- lapply(1:4, function(q) flexmix(wine_data ~ 1, model =
    FLXMCfactanal(factors = q, nstart = 3),
    cluster = posterior(wine_fl_diag)))
  sapply(wine_fl_fact, logLik)
## FULL
  set.seed(123)
  wine_full <- initFlexmix(wine_data ~ 1, k = 3, nrep = 10,
    model = FLXMCmvnorm(diagonal = FALSE))
  logLik(wine_full)
## TRUE
  wine_true <- flexmix(wine_data ~ 1, cluster = wine$Class,
    model = FLXMCmvnorm(diagonal = FALSE))
  logLik(wine_true)
}
```

---

**FLXMCmvbinary**  
*FlexMix Binary Clustering Driver*

**Description**

This is a model driver for *flexmix* implementing model-based clustering of binary data.

**Usage**

```r
FLXMCmvbinary(formula = . ~ ., truncated = FALSE)
```

**Arguments**

- `formula`  
  A formula which is interpreted relative to the formula specified in the call to `flexmix` using `update.formula`. Only the left-hand side (response) of the formula is used. Default is to use the original `flexmix` model formula.

- `truncated`  
  logical, if TRUE the observations for the pattern with only zeros are missing and the truncated likelihood is optimized using an EM-algorithm.

**Details**

This model driver can be used to cluster binary data. The only parameter is the column-wise mean of the data, which equals the probability of observing a 1.
Value

FLXMCMvmbinary returns an object of class FLXMC.

Author(s)

Friedrich Leisch and Bettina Gruen

See Also

flexmix

Description

This is a model driver for flexmix implementing model-based clustering of a combination of binary and Gaussian data.

Usage

FLXMCMvcombi(formula = . ~ .)

Arguments

formula

A formula which is interpreted relative to the formula specified in the call to flexmix using update.formula. Only the left-hand side (response) of the formula is used. Default is to use the original flexmix model formula.

Details

This model driver can be used to cluster mixed-mode binary and Gaussian data. It checks which columns of a matrix contain only zero and ones, and does the same as FLXMCMvmbinary for them. For the remaining columns of the data matrix independent Gaussian distributions are used (same as FLXMCMvnorm with diagonal = FALSE. The same could be obtained by creating a corresponding list of two models for the respective columns, but FLXMCMvcombi does a better job in reporting parameters.

Value

FLXMCMvcombi returns an object of class FLXMC.

Author(s)

Friedrich Leisch
FLXMCmnorm

See Also

flexmix, FLXMCmvbinary, FLXMCmnorm

Examples

## create some artificial data
x1 <- cbind(rnorm(300),
            sample(0:1, 300, replace = TRUE, prob = c(0.25, 0.75)))
x2 <- cbind(rnorm(300, mean = 2, sd = 0.5),
            sample(0:1, 300, replace = TRUE, prob = c(0.75, 0.25)))
x <- rbind(x1, x2)

## fit the model
f1 <- flexmix(x ~ 1, k = 2, model = FLXMCmvcombi())
## should be similar to the original parameters
parameters(f1)
  table(clusters(f1), rep(1:2, c(300, 300)))

## a column with noise should not hurt too much
x <- cbind(x, rnorm(600))
f2 <- flexmix(x ~ 1, k = 2, model = FLXMCmvcombi())
parameters(f2)
  table(clusters(f2), rep(1:2, c(300, 300)))

---

FLXMCmnorm

**FlexMix Clustering Demo Driver**

**Description**

These are demo drivers for flexmix implementing model-based clustering of Gaussian data.

**Usage**

```r
FLXMCmnorm(formula = . ~ ., diagonal = TRUE)
FLXMCnorm1(formula = . ~ .)
```

**Arguments**

- `formula` A formula which is interpreted relative to the formula specified in the call to `flexmix` using `update.formula`. Only the left-hand side (response) of the formula is used. Default is to use the original `flexmix` model formula.
- `diagonal` If TRUE, then the covariance matrix of the components is restricted to diagonal matrices.

**Details**

This is mostly meant as a demo for FlexMix driver programming, you should also look at package mclust for real applications. FLXMCmnorm clusters multivariate data, FLXMCnorm1 univariate data. In the latter case smart initialization is important, see the example below.
**Value**

`FLXMCmvnorm` returns an object of class `FLXMC`.

**Author(s)**

Friedrich Leisch and Bettina Gruen

**References**


**See Also**

`flexmix`

**Examples**

```r
data("Nclus", package = "flexmix")
require("MASS")
eqscplot(Nclus)

## This model is wrong (one component has a non-diagonal cov matrix)
ex1 <- flexmix(Nclus ~ 1, k = 4, model = FLXMCmvnorm())
print(ex1)
plotEll(ex1, Nclus)

## True model, wrong number of components
ex2 <- flexmix(Nclus ~ 1, k = 6, model = FLXCMvnorm(diagonal = FALSE))
print(ex2)
plotEll(ex2, Nclus)

## Get parameters of first component
parameters(ex2, component = 1)

## Have a look at the posterior probabilities of 10 random observations
ok <- sample(1:nrow(Nclus), 10)
p <- posterior(ex2)[ok, ]

## The following two should be the same
max.col(p)
clusters(ex2)[ok]

## Now try the univariate case
plot(density(Nclus[, 1]))
ex3 <- flexmix(Nclus[, 1] ~ 1, cluster = cut(Nclus[, 1], 3),
              model = FLXMCnorm1())
ex3
```
FLXMcmvpois

parameters(ex3)

FLXMcmvpois   FlexMix Poisson Clustering Driver

Description
This is a model driver for flexmix implementing model-based clustering of Poisson distributed data.

Usage
FLXMcmvpois(formula = . ~ .)

Arguments
formula A formula which is interpreted relative to the formula specified in the call to flexmix using update.formula. Only the left-hand side (response) of the formula is used. Default is to use the original flexmix model formula.

Details
This can be used to cluster Poisson distributed data where given the component membership the variables are mutually independent.

Value
FLXMcmvpois returns an object of class FLXMC.

Author(s)
Friedrich Leisch and Bettina Gruen

See Also
flexmix
Description

Model driver for fitting mixtures of conditional logit models.

Usage

FLXMRcondlogit(formula = . ~ ., strata)

Arguments

- **formula**: A formula which is interpreted relative to the formula specified in the call to flexmix using update.formula. Default is to use the original flexmix model formula.
- **strata**: A formula which is interpreted such that no intercept is fitted and which allows to determine the variable indicating which observations are from the same stratum.

Details

The M-step is performed using coxph.fit.

Value

Returns an object of class FLXMRcondlogit.

Warning

To ensure identifiability repeated measurements are necessary. Sufficient conditions are given in Gruen and Leisch (2008).

Author(s)

Bettina Gruen

References


See Also

FLXMRmultinom
Examples

```r
if (require("mlogit")) {
  data("Catsup", package = "mlogit")
  ## To reduce the time needed for the example only a subset is used
  Catsup <- subset(Catsup, id %% 1:100)
  Catsup$experiment <- seq_len(nrow(Catsup))
  vnames <- c("display", "feature", "price")
  Catsup_long <-
    reshape(Catsup,
           idvar = c("id", "experiment"),
           times = c(paste("heinz", c(41, 32, 28), sep = ""),
                     "hunts32"),
           timevar = "brand",
           varying = matrix(colnames(Catsup)[2:13], nrow = 3, byrow = TRUE),
           v.names = vnames,
           direction = "long")
  Catsup_long$selected <- with(Catsup_long, choice == brand)
  Catsup_long <- Catsup_long[, c("id", "selected", "experiment", vnames, "brand")]
  Catsup_long$brand <- relevel(factor(Catsup_long$brand), "hunts32")
  set.seed(0808)
  flx1 <- flexmix(selected ~ display + feature + price + brand | id,
                  model = FLXMRcondlogit(strata = ~ experiment),
                  data = Catsup_long, k = 1)
}
```

---

**FLXMRglm**

*FlexMix Interface to Generalized Linear Models*

**Description**

This is the main driver for FlexMix interfacing the **glm** family of models.

**Usage**

```r
FLXMRglm(formula = ~ ., 
          family = c("gaussian", "binomial", "poisson", "Gamma"),
          offset = NULL)
```

**Arguments**

- **formula**: A formula which is interpreted relative to the formula specified in the call to `flexmix` using `update.formula`. Default is to use the original `flexmix` model formula.
- **family**: A character string naming a **glm** family function.
- **offset**: This can be used to specify an *a priori* known component to be included in the linear predictor during fitting.
Details

See flexmix for examples.

Value

Returns an object of class FLXMRglm.

Author(s)

Friedrich Leisch and Bettina Gruen

References

Friedrich Leisch. FlexMix: A general framework for finite mixture models and latent class regres-


See Also

flexmix, glm

---

FLXMRglmfix

FlexMix Interface to GLMs with Fixed Coefficients

Description

This implements a driver for FlexMix which interfaces the glm family of models and where it is

desirable to specify fixed (constant) or nested varying coefficients or to ensure that in the Gaussian

case the variance estimate is equal for all components.

Usage

FLXMRglmfix(formula = ~ ., fixed = ~0, varFix = FALSE, nested = NULL,

family = c("gaussian", "binomial", "poisson", "Gamma"),

offset = NULL)

Arguments

formula A formula which is interpreted relative to the formula specified in the call to

flexmix using update.formula. Default is to use the original flexmix model

formula.

fixed A formula which specifies the additional regressors for the fixed (constant) co-

efficients.

varFix A logical indicating if the variance estimate for Gaussian components should be

constrained to be equal for all components. It can be also a vector specifying the

number of components with equal variance.

nested An object of class FLXnested or a list specifying the nested structure.
family A character string naming a glm family function.

offset This can be used to specify an a priori known component to be included in the linear predictor during fitting.

Value

Returns an object of class FLXMRglmfix.

Author(s)

Friedrich Leisch and Bettina Gruen

See Also

FLXMRglm

Examples

```r
data("NPreg", package = "flexmix")
ex <- flexmix(yn ~ x | id2, data = NPreg, k = 2,
cluster = NPreg$class,
model = FLXMRglm(yn ~ . + I(x^2)))
ex.fix <- flexmix(yn ~ x | id2, data = NPreg,
cluster = posterior(ex),
model = FLXMRglmfix(nested = list(k = c(1, 1),
formula = c(~0, ~I(x^2)))))
summary(refit(ex))
## Not run:
summary(refit(ex.fix))

## End(Not run)
```

---

**FLXMRglmnet**  
*FlexMix Interface for Adaptive Lasso / Elastic Net with GLMs*

**Description**

This is a driver which allows fitting of mixtures of GLMs where the coefficients are penalized using the (adaptive) lasso or the elastic net by reusing functionality from package *glmnet*.

**Usage**

```r
FLXMRglmnet(formula = . ~ ., family = c("gaussian", "binomial", "poisson"),
adaptive = TRUE, select = TRUE, offset = NULL, ...)
```
Arguments

- **formula**: A formula which is interpreted relative to the formula specified in the call to `flexmix` using `update.formula`. Default is to use the original `flexmix` model formula.

- **family**: A character string naming a `glm` family function.

- **adaptive**: A logical indicating if the adaptive lasso should be used. By default equal to `TRUE`.

- **select**: A logical vector indicating which variables in the model matrix should be included in the penalized part. By default equal to `TRUE` implying that all variables are penalized.

- **offset**: This can be used to specify an *a priori* known component to be included in the linear predictor during fitting.

- **...**: Additional arguments to be passed to `cv.glmnet` fitter.

Details

Some care is needed to ensure convergence of the algorithm, which is computationally more challenging than a standard EM. In the proposed method, not only are cluster allocations identified and component parameters estimated as commonly done in mixture models, but there is also variable selection via penalized regression using $k$-fold cross-validation to choose the penalty parameter. For the algorithm to converge, it is necessary that the same cross-validation partitioning be used across the EM iterations, i.e., the subsamples for cross-validation must be defined at the beginning. This is accomplished using the `foldid` option as an additional parameter to be passed to `cv.glmnet` (see `glmnet` package documentation).

Value

Returns an object of class `FLXMRglm`.

Author(s)

Frederic Mortier and Nicolas Picard.

References


See Also

`FLXMRglm`
Examples

```r
set.seed(12)
p <- 10
beta <- matrix(0, nrow = p + 1, ncol = 2)
beta[,1] <- c(-1, 1)
beta[cbind(c(5, 10), c(1, 2))] <- 1

nobs <- 100
X <- matrix(rnorm(nobs * p), nobs, p)
mu <- cbind(1, X) %*% beta
z <- sample(1:ncol(beta), nobs, replace = TRUE)
y <- mu[cbind(1:nobs, z)] + rnorm(nobs)
data <- data.frame(y, X)
## The maximum number of iterations is reduced to
## avoid a long running time.
fo <- sample(rep(seq(10), length = nrow(data)))
ex1 <- flexmix(y ~ ., data = data, k = 2, cluster = z,
model = FLXMRglmnet(foldid = fo),
control = list(iter.max = 2))
parameters(ex1)
```

FLXMRlmer

**FlexMix Interface to Linear Mixed Models**

**Description**

This is a driver which allows fitting of mixtures of linear models with random effects.

**Usage**

```r
FLXMRlmm(formula = . ~ ., random, lm.fit = c("lm.wfit",
"smooth.spline"), varfix = c(Random = FALSE, Residual = FALSE), ...)
FLXMRlmer(formula = . ~ ., random, weighted = TRUE,
control = list(), eps = .Machine$double.eps)
```

**Arguments**

- `formula`: A formula which is interpreted relative to the formula specified in the call to `flexmix` using `update.formula`. Default is to use the original `flexmix` model formula.
- `random`: A formula for specifying the random effects.
- `weighted`: A logical indicating if the model should be estimated with weighted ML.
- `control`: A list of control parameters. See `lmer` for details.
- `eps`: Observations with a component-specific posterior smaller than `eps` are omitted in the M-step for this component.
lm.fit  A character string indicating if the coefficients should be fitted using either a linear model or the function smooth.spline

varFix  Named logical vector of length 2 indicating if the variance of the random effects and the residuals are fixed over the components.

...  Additional arguments to be passed to smooth.spline.

Details

FLXMRlmm allows only one random effect. FLXMRlmer allows an arbitrary number of random effects if weighted = FALSE; a certain structure of the model matrix of the random effects has to be given for weighted ML estimation, i.e. where weighted = TRUE.

Value

Returns an object of class FLXMRlmer and FLXMRlmm.

Warning

For FLXMRlmer the weighted ML estimation is only correct if the covariate matrix of the random effects is the same for each observation. By default weighted ML estimation is made and the condition on the covariate matrix of the random effects is checked. If this fails, only estimation with weighted = FALSE is possible which will maximize the classification likelihood.

Author(s)

Bettina Gruen

Examples

id <- rep(1:50, each = 10)
x <- rep(1:10, 50)
sample <- data.frame(y = rep(rnorm(unique(id)/2, 0, c(5, 2), each = 10) +
                    rnorm(length(id), rep(c(3, 8), each = 10)) +
                    rep(c(0, 3), each = 10) * x,
                    x = x,
                    id = factor(id))
fitted <- flexmix(~1 | id, k = 2, model = FLXMRlmm(y ~ x, random = ~ 1),
data = sample, control = list(tolerance = 1e-3),
cluster = rep(rep(1:2, each = 10), 25))
parameters(fitted)
Usage

```r
FLXRnlmmc(formula = . ~ ., random, censored, varFix, eps = 10^-6, ...)
```

Arguments

- **formula**: A formula which is interpreted relative to the formula specified in the call to `flexmix` using `update.formula`. Default is to use the original `flexmix` model formula.
- **random**: A formula for specifying the random effects. If missing no random effects are fitted.
- **varFix**: If random effects are specified a named logical vector of length 2 indicating if the variance of the random effects and the residuals are fixed over the components. Otherwise a logical indicating if the variance of the residuals are fixed over the components.
- **censored**: A formula for specifying the censoring variable.
- **eps**: Observations with an a-posteriori probability smaller or equal to `eps` are omitted in the M-step.
- **...**: Additional arguments to be passed to `lm.wfit`.

Value

Returns an object of class `FLXRnlmmc`, `FLXRnlmcfix`, `FLXRnlmc` or `FLXRnlmcfix`.

Author(s)

Bettina Gruen

Description

This is a driver which allows fitting of mixtures of GAMs.

Usage

```r
FLXRmgcv(formula = . ~ ., family = c("gaussian", "binomial", "poisson"),
          offset = NULL, control = NULL, optimizer = c("outer", "newton"),
          in.out = NULL, eps = .Machine$double.eps, ...)
```
Arguments

formula A formula which is interpreted relative to the formula specified in the call to flexmix using update.formula. Default is to use the original flexmix model formula.

family A character string naming a glm family function.

offset This can be used to specify an a priori known component to be included in the linear predictor during fitting.

control A list of fit control parameters returned by gam.control.

optimizer An array specifying the numerical optimization method to use to optimize the smoothing parameter estimation criterion; for more details see gam.

in.out Optional list for initializing outer iteration; for more details see gam.

eps Observations with an a-posteriori probability smaller or equal to eps are omitted in the M-step.

Additional arguments to be passed to the GAM fitter.

Value

Returns an object of class FLXMRmgcv.

Author(s)

Bettina Gruen

See Also

FLXMRglm

Examples

set.seed(2012)
x <- seq(0, 1, length.out = 100)
z <- sample(0:1, length(x), replace = TRUE)
y <- rnorm(length(x), ifelse(z, 5 * sin(x * 2 * pi), 10 * x - 5))
fitted_model <- flexmix(y ~ s(x), model = FLXMRmgcv(),
                        cluster = z + 1,
                        control = list(tolerance = 10^-3))
plot(y ~ x, col = clusters(fitted_model))
matplot(x, fitted(fitted_model), type = "l", add = TRUE)
FLXMRmultinom

FlexMix Interface to Multinomial Logit Models

Description

Model driver for fitting mixtures of multinomial logit models.

Usage

FLXMRmultinom(formula = . ~ ., ...)

Arguments

formula            A formula which is interpreted relative to the formula specified in the call to
                   flexmix using update.formula. Default is to use the original flexmix model
                   formula.
...               Additional arguments to be passed to nnet.default.

Details

The M-step is performed using nnet.default.

Value

Returns an object of class FLXMRmultinom.

Warning

To ensure identifiability repeated measurements are necessary. Sufficient conditions are given in
Gruen and Leisch (2008).

Author(s)

Bettina Gruen

References

Bettina Gruen and Friedrich Leisch. Identifiability of finite mixtures of multinomial logit models
with varying and fixed effects. Journal of Classification, 25, 225–247, 2008.

See Also

FLXMRcondlogit
FLXMRrobglm

FlexMix Driver for Robust Estimation of Generalized Linear Models

Description

This driver adds a noise component to the mixture model which can be used to model background noise in the data. See the Compstat paper Leisch (2008) cited below for details.

Usage

FLXMRrobglm(formula = ~ ., family = c("gaussian", "poisson"), bgw = FALSE, ...)

Arguments

- `formula`: A formula which is interpreted relative to the formula specified in the call to `flexmix` using `update.formula`. Default is to use the original `flexmix` model formula.
- `family`: A character string naming a `glm` family function.
- `bgw`: Logical, controls whether the parameters of the background component are fixed to multiples of location and scale of the complete data (the default), or estimated by EM with normal weights for the background (bgw = TRUE).
- `...`: passed to `FLXMRglm`

Value

Returns an object of class FLXMRrobglm.

Note

The implementation of this model class is currently under development, and some methods like `refit` are still missing.

Author(s)

Friedrich Leisch and Bettina Gruen

References

Examples

```r
## Example from Compstat paper, see paper for detailed explanation:
data("NPreg", package = "flexmix")
DATA <- NPreg[, 1:2]
set.seed(3)
DATA2 <- rbind(DATA, cbind(x = rrunif(3), y = 50 + rrunif(3)))

## Estimation without (f2) and with (f3) background component
f2 <- flexmix(y ~ x + I(x^2), data = DATA2, k = 2)
f3 <- flexmix(y ~ x + I(x^2), data = DATA2, k = 3,
              model = FLXMRrobglm(),
              control = list(minprior = 0))

## Predict on new data for plots
x <- seq(-5, 15, by = .1)
y2 <- predict(f2, newdata = data.frame(x = x))
y3 <- predict(f3, newdata = data.frame(x = x))

## f2 was estimated without background component:
plot(y ~ x, data = DATA2, pch = clusters(f2), col = clusters(f2))
lines(x, y2$Comp.1, col = 1)
lines(x, y2$Comp.2, col = 2)

## f3 is with background component:
plot(y ~ x, data = DATA2, pch = 4 - clusters(f3),
     col = 4 - clusters(f3))
lines(x, y3$Comp.2, col = 2)
lines(x, y3$Comp.3, col = 1)
```

FLXMRziglm  

*FlexMix Interface to Zero Inflated Generalized Linear Models*

**Description**

This is a driver which allows fitting of zero inflated poisson and binomial models.

**Usage**

```r
FLXMRziglm(formula = . ~ ., family = c("binomial", "poisson"), ...)
```

**Arguments**

- `formula`: A formula which is interpreted relative to the formula specified in the call to `flexmix` using `update.formula`. Default is to use the original `flexmix` model formula.
- `family`: A character string naming a `glm` family function.
- `...`: Passed to `FLXMRglm`
Value

Returns an object of class FLXMZiglm.

Note

In fact this only approximates zero inflated models by fixing the coefficient of the intercept at -Inf and the other coefficients at zero for the first component.

Author(s)

Friedrich Leisch and Bettina Gruen

Examples

```r
data("dmft", package = "flexmix")
Model <- FLXMZiglm(family = "poisson")
Fitted <- flexmix(End ~ log(Begin + 0.5) + Gender + Ethnic + Treatment, 
  model = Model, k = 2, data = dmft, 
  control = list(minprior = 0.01))
summary(refit(Fitted))
```

FLXnested-class  Class "FLXnested"

Description

Specification of nesting structure for regression coefficients.

Objects from the Class

Objects can be created by calls of the form new("FLXnested", formula, k, ...). In addition, named lists can be coerced to FLXnested objects, names are completed if unique.

Slots

- `formula`: Object of class "list" containing the formula for determining the model matrix for each nested parameter.
- `k`: Object of class "numeric" specifying the number of components in each group.

Author(s)

Friedrich Leisch and Bettina Gruen
FLXP

 создан, нумерация предложения от 1

Description

Creator functions for the concomitant variable model. FLXPconstant specifies constant priors and FLXPMultinom multinomial logit models for the priors.

Usage

FLXPconstant()
FLXPMultinom(formula = ~1)

Arguments

formula A formula for determining the model matrix of the concomitant variables.

Details

FLXPMultinom uses nnet.default from nnet to fit the multinomial logit model.

Value

Object of class FLXP. FLXPMultinom returns an object of class FLXPMultinom which extends class FLXP directly and is used for method dispatching.

Author(s)

Friedrich Leisch and Bettina Gruen

FLXP-class

Class "FLXP"

Description

Concomitant model class.

Objects from the Class

Objects can be created by calls of the form new("FLXP", ...), typically inside driver functions like FLXPConstant or FLXPMultinom.
Slots

name: Character string used in print methods.

formula: Formula describing the model.

x: Model matrix.

fit: Function returning the fitted prior probabilities.

refit: Function returning the fitted concomitant model.

coef: Matrix containing the fitted parameters.

df: Function for determining the number of degrees of freedom used.

Author(s)

Friedrich Leisch and Bettina Gruen

---

**group**

*Extract Grouping Variable*

Description

Extract grouping variable for all observations.

Usage

```r
## S4 method for signature 'flexmix'
group(object)
## S4 method for signature 'FLXM'
group(object)
## S4 method for signature 'FLXMRglmfix'
group(object)
```

Arguments

- `object`: an object of class `flexmix`.

Author(s)

Bettina Gruen
ICL

Integrated Completed Likelihood Criterion

Description
Compute the Integrated Completed Likelihood criterion for model selection.

Usage

```r
# S4 method for signature 'flexmix'
ICL(object, ...)
# S4 method for signature 'stepFlexmix'
ICL(object, ...)
```

Arguments

- **object**: see Methods section below
- **...**: Some methods for this generic function may take additional, optional arguments. At present none do.

Value

Returns a numeric vector with the corresponding ICL value(s).

Methods

- **object = "flexmix"**: Compute the ICL of a `flexmix` object.
- **object = "stepFlexmix"**: Compute the ICL of all models contained in the `stepFlexmix` object.

Author(s)

Friedrich Leisch and Bettina Gruen

References


Examples

```r
data("NPreg", package = "flexmix")
ex1 <- flexmix(y ~ x + I(x^2), data = NPreg, k = 2)
ICL(ex1)
```
KLdiv  

**Kullback-Leibler Divergence**

### Description
Estimate the Kullback-Leibler divergence of several distributions.

### Usage

```r
## S4 method for signature 'matrix'
KLdiv(object, eps = 10^-4, overlap = TRUE, ...)
## S4 method for signature 'flexmix'
KLdiv(object, method = c("continuous", "discrete"), ...)
```

### Arguments
- **object**: See Methods section below.
- **method**: The method to be used; "continuous" determines the Kullback-Leibler divergence between the unweighted theoretical component distributions and the unweighted posterior probabilities at the observed points are used by "discrete".
- **eps**: Probabilities below this threshold are replaced by this threshold for numerical stability.
- **overlap**: Logical, do not determine the KL divergence for those pairs where for each point at least one of the densities has a value smaller than eps.
- ... Passed to the matrix method.

### Details
Estimates

\[
\int f(x)(\log f(x) - \log g(x))dx
\]

for distributions with densities \(f()\) and \(g()\).

### Value
A matrix of KL divergences where the rows correspond to using the respective distribution as \(f()\) in the formula above.

### Methods
- **object = "matrix"**: Takes as input a matrix of density values with one row per observation and one column per distribution.
- **object = "flexmix"**: Returns the Kullback-Leibler divergence of the mixture components.
Note

The density functions are modified to have equal support. A weight of at least \( \varepsilon \) is given to each observation point for the modified densities.

Author(s)

Friedrich Leisch and Bettina Gruen

References


Examples

```r
## Gaussian and Student t are much closer to each other than
## to the uniform:

x <- seq(-3, 3, length = 200)
y <- cbind(u = dunif(x), n = dnorm(x), t = dt(x, df = 10))

matplot(x, y, type = "l")

if (require("mlbench")) {
  set.seed(25056)
x <- mlbench.smiley()$x
model1 <- flexmix(x ~ 1, k = 9, model = FLXmclust(diag = FALSE),
  control = list(minprior = 0))
plotEll(model1, x)
KLDiv(model1)
}
```

Lapply-methods

Methods for Function `Lapply`

Description

Apply a function to each component of a finite mixture

Usage

```r
## S4 method for signature 'FLXmstep'
Lapply(object, FUN, model = 1, component = TRUE, ...)
```
Arguments

object S4 class object.
FUN The function to be applied.
model The model (for a multivariate response) that shall be used.
component Index vector for selecting the components.
... Optional arguments to FUN.

Details

FUN is found by a call to match.fun and typically is specified as a function or a symbol (e.g. a back-quoted name) or a character string specifying a function to be searched for from the environment of the call to lapply.

Value

A list of the length equal to the number of components specified is returned, each element of which is the result of applying FUN to the specified component of the refitted mixture model.

Methods

object = FLXRmstep: Apply a function to each component of a refitted flexmix object using method = "mstep".

Author(s)

Friedrich Leisch and Bettina Gruen

Examples

data("NPreg", package = "flexmix")
ex2 <- flexmix(yn ~ x, data = NPreg, k = 2, model = list(FLXMRglm(yn ~
. + I(x^2)), FLXMRglm(yp ~ ., family = "poisson")))
ex2r <- refit(ex2, method = "mstep")
lapply(ex2r, "vcov", 2)

logLik-methods Methods for Function logLik in Package flexmix

Description

Evaluate the log-likelihood. This function is defined as an S4 generic in the stats4 package.

Methods

object = flexmix Evaluate the log-likelihood of an flexmix object
Description

For a 22-centre trial the number of responses and the total number of patients is reported for the control group and the group receiving a new drug.

Usage

data("Mehta")

Format

A data frame with 44 observations on the following 4 variables.

- **Response**: Number of responses.
- **Total**: Total number of observations.
- **Drug**: A factor indicating treatment with levels `New` and `Control`.
- **Site**: A factor indicating the site/centre.

Source


References


Examples

data("Mehta", package = "flexmix")
mehtaMix <- initFlexmix(cbind(Response, Total~Response) ~ 1|Site,
                         data = Mehta, nrep = 5, k = 3,
                         model = FLXMIXGLMfix(family = "binomial",
                                             fixed = ~ Drug),
                         control = list(minprior = 0.04))
**Artificial Example for Normal Regression**

**Description**

A simple artificial regression example with 3 latent classes, two independent variables, one concomitant variable and a dependent variable which follows a Gaussian distribution.

**Usage**

```r
data("NregFix")
```

**Format**

A data frame with 200 observations on the following 5 variables.

- **x1**: Independent variable: numeric variable.
- **x2**: Independent variable: a factor with two levels: 0 and 1.
- **w**: Concomitant variable: a factor with two levels: 0 and 1.
- **y**: Dependent variable.
- **class**: Latent class memberships.

**Examples**

```r
data("NregFix", package = "flexmix")
library("lattice")
xyplot(y ~ x1 | x2 * w, data = NregFix, groups = class)
Model <- FLXMRglmfix(~ 1, fixed = ~ x2,
                     nested = list(k = c(2, 1),
                     formula = c(~x1, ~0)))
fittedModel <- initFlexmix(y ~ 1, model = Model, data = NregFix, k = 3,
                          concomitant = FLXPmultinom(~ w), nrep = 5)
fittedModel
```

---

**Patents and R&D Spending**

**Description**

Number of patents, R&D spending and sales in millions of dollar for 70 pharmaceutical and biomedical companies in 1976.

**Usage**

```r
data("patent")
```
**Format**

A data frame with 70 observations on the following 4 variables.

- **Company**: Name of company.
- **Patents**: Number of patents.
- **RDS**: R&D spending per sales.
- **lgRD**: Logarithmized R&D spendings (in millions of dollars).

**Details**

The data is taken from the National Bureau of Economic Research R&D Masterfile.

**Source**


**References**


**Examples**

```r
data("patent", package = "flexmix")
patentMix <- initFlexmix(Patents ~ lgRD, k = 3,
  model = FLXMRglm(family = "poisson"),
  concomitant = FLXPMultinom(~RDS),
  nrep = 5, data = patent)
plot(Patents ~ lgRD, data = patent,
  pch = as.character(clusters(patentMix)))
ordering <- order(patent$lgRD)
apply(fitted(patentMix), 2, function(y)
  lines(sort(patent$lgRD), y[ordering]))
```

---

**Description**

The `plot` method for `flexmix-class` objects gives a rootogram or histogram of the posterior probabilities.

**Usage**

```r
## S4 method for signature 'flexmix,missing'
plot(x, y, mark = NULL, markcol = NULL,
  col = NULL, eps = 1e-4, root = TRUE, ylim = TRUE, main = NULL, xlab = "",
  ylab = "", as.table = TRUE, endpoints = c(-0.04, 1.04), ...)
```
Arguments

x             An object of class "flexmix".
y             Not used.
mark         Integer: mark posteriors of this component.
markcol      Color used for marking components.
col          Color used for the bars.
eps          Posteriors smaller than eps are ignored.
root         If TRUE, a rootogram of the posterior probabilities is drawn, otherwise a standard histogram.
ylim         A logical value or a numeric vector of length 2. If TRUE, the y axes of all rootograms are aligned to have the same limits, if FALSE each y axis is scaled separately. If a numeric vector is specified it is used as usual.
main         Main title of the plot.
xlab         Label of x-axis.
ylab         Label of y-axis.
as.table     Logical that controls the order in which panels should be plotted: if FALSE (the default), panels are drawn left to right, bottom to top (as in a graph); if TRUE, left to right, top to bottom.
endpoints    Vector of length 2 indicating the range of x-values that is to be covered by the histogram. This applies only when breaks is unspecified. In do.breaks, this specifies the interval that is to be divided up.
...          Further graphical parameters for the lattice function histogram.

Details

For each mixture component a rootogram or histogram of the posterior probabilities of all observations is drawn. Rootograms are very similar to histograms, the only difference is that the height of the bars correspond to square roots of counts rather than the counts themselves, hence low counts are more visible and peaks less emphasized. Please note that the y-axis denotes the number of observations in each bar in any case.

Usually in each component a lot of observations have posteriors close to zero, resulting in a high count for the corresponding bin in the rootogram which obscures the information in the other bins. To avoid this problem, all probabilities with a posterior below eps are ignored.

A peak at probability one indicates that a mixture component is well seperated from the other components, while no peak at one and/or significant mass in the middle of the unit interval indicates overlap with other components.

Author(s)

Friedrich Leisch and Bettina Gruen
References


### plotElli

*Plot Confidence Ellipses for FLXMcmvnorm Results*

**Description**

Plot 50% and 95% confidence ellipses for mixtures of Gaussians fitted using FLXMcmvnorm.

**Usage**

```r
plotElli(object, data, which = 1:2, model = 1, project = NULL, points = TRUE,
        eqscale = TRUE, col = NULL, number = TRUE, cex = 1.5, numcol = "black",
        pch = NULL, ...)
```

**Arguments**

- `object`: An object of class `flexmix` with a fitted FLXMcmvnorm model.
- `data`: The response variable in a data frame or as a matrix.
- `which`: Index numbers of dimensions of (projected) input space to plot.
- `model`: The model (for a multivariate response) that shall be plotted.
- `project`: Projection object, currently only the result of `prcomp` is supported.
- `points`: Logical, shall data points be plotted?
- `eqscale`: Logical, plot using `eqscplot`?
- `number`: Logical, plot number labels at cluster centers?
- `cex`, `numcol`: Size and color of number labels.
- `pch`, `col`, ... Graphical parameters.

**Author(s)**

Friedrich Leisch and Bettina Gruen

**See Also**

FLXMcmvnorm
### posterior

**Determine Cluster Membership and Posterior Probabilities**

**Description**

Determine posterior probabilities or cluster memberships for a fitted `flexmix` or unfitted `FLXdist` model.

**Usage**

```r
## S4 method for signature 'flexmix,missing'
posterior(object, newdata, unscaled = FALSE, ...)
## S4 method for signature 'FLXdist,listOrdata.frame'
posterior(object, newdata, unscaled = FALSE, ...)
## S4 method for signature 'flexmix,missing'
clusters(object, newdata, ...)
## S4 method for signature 'FLXdist,ANY'
clusters(object, newdata, ...)
```

**Arguments**

- `object` An object of class "flexmix" or "FLXdist".
- `newdata` Data frame or list containing new data. If missing the posteriors of the original observations are returned.
- `unscaled` Logical, if TRUE the component-specific likelihoods are returned.
- `...` Currently not used.

**Author(s)**

Friedrich Leisch and Bettina Gruen

---

### refit-methods

**Refit a Fitted Model**

**Description**

Refits an estimated `flexmix` model to obtain additional information like coefficient significance p-values for GLM regression.
Usage

```r
## S4 method for signature 'flexmix'
refit(object, newdata, method = c("optim", "mstep"), ...)
## S4 method for signature 'FLXRoptim'
summary(object, model = 1, which = c("model", "concomitant"), ...)
## S4 method for signature 'FLXRmstep'
summary(object, model = 1, which = c("model", "concomitant"), ...)

## S4 method for signature 'FLXRoptim,missing'
plot(x, y, model = 1, which = c("model", "concomitant"),
     bycluster = TRUE, alpha = 0.05, components, labels = NULL,
     significance = FALSE, xlab = NULL, ylab = NULL, ci = TRUE,
     scales = list(), as.table = TRUE, horizontal = TRUE, ...)
```

Arguments

- **object**: An object of class "flexmix"
- **newdata**: Optional new data.
- **method**: Specifies if the variance covariance matrix is determined using optim or if the posteriors are assumed as given and an M-step is performed.
- **model**: The model (for a multivariate response) that shall be used.
- **which**: Specifies if a component specific model or the concomitant variable model is used.
- **x**: An object of class "FLXRoptim"
- **y**: Missing object.
- **bycluster**: A logical if the parameters should be group by cluster or by variable.
- **alpha**: Numeric indicating the significance level.
- **components**: Numeric vector specifying which components are plotted. The default is to plot all components.
- **labels**: Character vector specifying the variable names used.
- **significance**: A logical indicating if non-significant coefficients are shaded in a lighter grey.
- **xlab**: String for the x-axis label.
- **ylab**: String for the y-axis label.
- **ci**: A logical indicating if significant and insignificant parameter estimates are shaded differently.
- **scales**: See argument of the same name for function `xyplot`.
- **as.table**: See arguments of the same name for function `xyplot`.
- **horizontal**: See arguments of the same name for function `xyplot`.
- **...**: Currently not used
Details

The refit method for FLXMrglm models in combination with the summary method can be used to obtain the usual tests for significance of coefficients. Note that the tests are valid only if flexmix returned the maximum likelihood estimator of the parameters. If refit is used with method = "mstep" for these component specific models the returned object contains a glm object for each component where the elements model which is the model frame and data which contains the original dataset are missing.

Value

An object inheriting form class FLXR is returned. For the method using optim the object has class FLXROptim and for the M-step method it has class FLXRmstep. Both classes give similar results for their summary methods. Objects of class FLXROptim have their own plot method. lapply can be used to further analyse the refitted component specific models of objects of class FLXRmstep.

Warning

For method = "mstep" the standard deviations are determined separately for each of the components using the a-posteriori probabilities as weights without accounting for the fact that the components have been simultaneously estimated. The derived standard deviations are hence approximative and should only be used in an exploratory way, as they are underestimating the uncertainty given that the missing information of the component memberships are replaced by the expected values.

The newdata argument can only be specified when using method = "mstep" for refitting FLXMrglm components. A variant of glm for weighted ML estimation is used for fitting the components and full glm objects are returned. Please note that in this case the data and the model frame are stored for each component which can significantly increase the object size.

Author(s)

Friedrich Leisch and Bettina Gruen

References


Examples

data("NPreg", package = "flexmix")
ex1 <- flexmix(yn ~ x + I(x^2), data = NPreg, k = 2)
ex1r <- refit(ex1)

## in one component all coefficients should be highly significant,
## in the other component only the linear term
summary(ex1r)
**relabel**  

**Relabel the Components**

**Description**

The components are sorted by the value of one of the parameters or according to an integer vector containing the permutation of the numbers from 1 to the number of components.

**Usage**

```r
relabel(object, by, ...)  
## S4 method for signature 'FLXdist,character'
relabel(object, by, which = NULL, ...)
```

**Arguments**

- **object**
  - An object of class "flexmix".
- **by**
  - If a character vector, it needs to be one of "prior", "model", "concomitant" indicating if the parameter should be from the component-specific or the concomitant variable model. If an integer vector it indicates how the components should be sorted. If missing, the components are sorted by component size.
- **which**
  - Name (or unique substring) of a parameter if by is equal to "model" or "concomitant".
- **...**
  - Currently not used.

**Author(s)**

Friedrich Leisch and Bettina Gruen

**Examples**

```r
set.seed(123)
beta <- matrix(1:16, ncol = 4)
beta
df1 <- ExLinear(beta, n = 100, sd = .5)
f1 <- flexmix(y ~ ., data = df1, k = 4)

## There was label switching, parameters are not in the same order
## as in beta:
round(parameters(f1))

betas <- rbind(beta, .5)
betas

## This makes no sense:
summary(abs(as.vector(betas-parameters(f1)))))

## We relabel the components by sorting the coefficients of x1:
```

rflexmix <- relabel(f1, by = "model", which = "x1")
round(parameters(r1))

### Now we can easily compare the fit with the true parameters:
summary(abs(as.vector(betas-parameters(r1)))))

---

**rflexmix**  
*Random Number Generator for Finite Mixtures*

**Description**

Given a finite mixture model generate random numbers from it.

**Usage**

rflexmix(object, newdata, ...)

**Arguments**

- `object`: A fitted finite mixture model of class `flexmix` or an unfitted of class `FLXdist`.
- `newdata`: Optionally, a data frame in which to look for variables with which to predict or an integer specifying the number of random draws for model-based clustering. If omitted, the data to which the model was fitted is used.
- `...`: Further arguments to be passed to or from methods.

**Details**

`rflexmix` provides the creation of the model matrix for new data and the sampling of the cluster memberships. The sampling of the component distributions given the classification is done by calling `rFLXm`. This step has to be provided for the different model classes.

**Value**

A list with components

- `y`: Random sample
- `group`: Grouping factor
- `class`: Class membership

**Author(s)**

Bettina Gruen

**Examples**

eexample(flexmix)
sample <- rflexmix(ex1)
Description

Data on Ames Salmonella reverse mutagenicity assay.

Usage

data("salmonellaTA98")

Format

This data frame contains the following columns:

- **x**: Dose levels of quinoline.
- **y**: Numbers of revertant colonies of TA98 Salmonella observed on each of three replicate plates tested at each of six dose levels of quinoline diameter.

Details

This data set is taken from package *dispmod* provided by Luca Scrucca.

Source


References


Examples

data("salmonellaTA98", package = "flexmix")
salmonMix <- initFlexmix(y ~ 1,
  data = salmonellaTA98,
  model = FLXMRglmfix(family = "poisson",
    fixed = ~ x + log(x + 10)),
  k = 2, nrep = 5)
salmonMix.pr <- predict(salmonMix, newdata = salmonellaTA98)
plot(y ~ x, data = salmonellaTA98,
  pch = as.character(clusters(salmonMix)),
  ylim = range(c(salmonellaTA98$y, unlist(salmonMix.pr))))
for (i in 1:2) lines(salmonellaTA98$x, salmonMix.pr[[i]], lty = i)
**Description**

Data from a clinical trial where the effect of intravenous gamma-globulin on suppression of epileptic seizures is studied. Daily observations for a period of 140 days on one patient are given, where the first 27 days are a baseline period without treatment, the remaining 113 days are the treatment period.

**Usage**

data("seizure")

**Format**

A data frame with 140 observations on the following 4 variables.

Seizures A numeric vector, daily counts of epileptic seizures.

Hours A numeric vector, hours of daily parental observation.

Treatment A factor with levels No and Yes.

Day A numeric vector.

**Source**


**References**


**Examples**

data("seizure", package = "flexmix")
plot(Seizures/Hours ~ Day, col = as.integer(Treatment),
    pch = as.integer(Treatment), data = seizure)
abline(v = 27.5, lty = 2, col = "grey")
legend(140, 9, c("Baseline", "Treatment"),
    pch = 1:2, col = 1:2, xjust = 1, yjust = 1)
set.seed(123)

## The model presented in the Wang et al paper: two components for
## "good" and "bad" days, respectively, each a Poisson GLM with hours of
## parental observation as offset
```r
seizMix <- flexmix(Seizures ~ Treatment * log(Day),
  data = seizure, k = 2,
  model = FLXMRglm(family = "poisson",
      offset = log(seizure$Hours)))

summary(seizMix)
summary(refit(seizMix))

matplot(seizure$Day, fitted(seizMix)/seizure$Hours, type = "l",
  add = TRUE, col = 3:4)
```

---

### stepFlexmix

#### Run FlexMix Repeatedly

**Description**

Runs `flexmix` repeatedly for different numbers of components and returns the maximum likelihood solution for each.

**Usage**

```r
initFlexmix(..., k, init = list(), control = list(), nrep = 3L,
  verbose = TRUE, drop = TRUE, unique = FALSE)
initMethod(name = c("tol.em", "cem.em", "sem.em"),
  step1 = list(tolerance = 10^-2),
  step2 = list(), control = list(), nrep = 3L)

stepFlexmix(..., k = NULL, nrep = 3, verbose = TRUE, drop = TRUE,
  unique = FALSE)
```

```r
# S4 method for signature 'stepFlexmix,missing'
plot(x, y, what = c("AIC", "BIC", "ICL"),
  xlab = NULL, ylab = NULL, legend = "topright", ...)
```

```r
# S4 method for signature 'stepFlexmix'
getModel(object, which = "BIC")
```

```r
# S4 method for signature 'stepFlexmix'
unique(x, incomparables = FALSE, ...)
```

**Arguments**

- `...` Passed to `flexmix` (or `matplot` in the plot method).
- `k` A vector of integers passed in turn to the `k` argument of `flexmix`.
- `init` An object of class "initMethod" or a named list where initMethod is called with it as arguments in addition to the control argument.
name
A character string indicating which initialization strategy should be employed:
short runs of EM followed by a long ("tol.em"), short runs of CEM followed
by a long EM run ("cem.em"), short runs of SEM followed by a long EM run
("sem.em").

step1
A named list which combined with the control argument is coercable to a
"FLXcontrol" object. This control setting is used for the short runs.

step2
A named list which combined with the control argument is coercable to a
"FLXcontrol" object. This control setting is used for the long run.

control
A named list which combined with the step1 or the step2 argument is coercable
to a "FLXcontrol" object.

nrep
For each value of \( k \) run \texttt{flexmix} \texttt{nrep} times and keep only the solution with
maximum likelihood. If \texttt{nrep} is set for the long run, it is ignored, because the
EM algorithm is deterministic using the best solution discovered in the short
runs for initialization.

verbose
If \texttt{true}, show progress information during computations.

drop
If \texttt{true} and \( k \) is of length 1, then a single \texttt{flexmix} object is returned instead of a
"stepFlexmix" object.

unique
If \texttt{true}, then \texttt{unique()} is called on the result, see below.

x, object
An object of class "stepFlexmix".

y
Not used.

what
Character vector naming information criteria to plot. Functions of the same
name must exist, which take a \texttt{stepFlexmix} object as input and return a numeric
vector like \texttt{aicLstepflexmixMmethod} (see examples below).

xlab,ylab
Graphical parameters.

legend
If not \texttt{false} and \texttt{what} contains more than 1 element, a legend is placed at the
specified location, see \texttt{legend} for details.

which
Number of model to get. If character, interpreted as number of components or
name of an information criterion.

incomparables
A vector of values that cannot be compared. Currently, \texttt{FALSE} is the only possible
value, meaning that all values can be compared.

Value
An object of class "stepFlexmix" containing the best models with respect to the log likelihood
for the different number of components in a slot if \texttt{length(k)>1}, else directly an object of class
"flexmix".

If \texttt{unique = FALSE}, then the resulting object contains one model per element of \( k \) (which is the
number of clusters the EM algorithm started with). If \texttt{unique = TRUE}, then the result is resorted
according to the number of clusters contained in the fitted models (which may be less than the
number with which the EM algorithm started), and only the maximum likelihood solution for each
number of fitted clusters is kept. This operation can also be done manually by calling \texttt{unique()} on
objects of class "stepFlexmix".
Author(s)
Friedrich Leisch and Bettina Gruen

References

Examples
```r
data("Nclus", package = "flexmix")

## try 2 times for k = 4
set.seed(511)
ex1 <- initFlexmix(Nclus-1, k = 4, model = FLXMCmvnorm(diagonal = FALSE),
                   nrep = 2)
ex1

## now 2 times each for k = 2:5, specify control parameter
ex2 <- initFlexmix(Nclus-1, k = 2:5, model = FLXMCmvnorm(diagonal = FALSE),
                   control = list(minprior = 0), nrep = 2)
ex2
plot(ex2)

## get BIC values
BIC(ex2)

## get smallest model
getModel(ex2, which = 1)

## get model with 3 components
getModel(ex2, which = "3")

## get model with smallest ICL (here same as for AIC and BIC: true k = 4)
getModel(ex2, which = "ICL")

## now 1 time each for k = 2:5, with larger minimum prior
ex3 <- initFlexmix(Nclus-1, k = 2:5,
                   model = FLXMCmvnorm(diagonal = FALSE),
                   control = list(minprior = 0.1), nrep = 1)
ex3

## keep only maximum likelihood solution for each unique number of
## fitted clusters:
unique(ex3)
```
Description

The data investigates whether the adult Tribolium species Castaneum has developed an evolutionary advantage to recognize and avoid eggs of their own species while foraging.

Usage

```r
data("tribolium")
```

Format

A data frame with 27 observations on the following 4 variables.

- Remaining: A numeric vector.
- Total: A numeric vector.
- Replicate: A factor with levels 1, 2, 3.
- Species: A factor with levels Castaneum Confusum Madens.

Details

Beetles of the genus Tribolium are cannibalistic in the sense that adults eat the eggs of their own species as well as those of closely related species. The experiment isolated a number of adult beetles of the same species and presented them with a vial of 150 eggs (50 of each type), the eggs being thoroughly mixed to ensure uniformity throughout the vial.

The data gives the consumption data for adult Castaneum species. It reports the number of Castaneum, Confusum and Madens eggs, respectively, that remain uneaten after two day exposure to the adult beetles. Replicates 1, 2, and 3 correspond to different occasions on which the experiment was conducted.

Source


Examples

```r
data("tribolium", package = "flexmix")
tribMix <- initFlexmix(cbind(Remaining, Total - Remaining) ~ Species,
                        k = 2, nrep = 5, data = tribolium,
                        model = FLXMRglm(family = "binomial"))
```
Trypanosome data from a dosage-response analysis to assess the proportion of organisms belonging to different populations. It is assumed that organisms belonging to different populations are indistinguishable other than in terms of their reaction to the stimulus.

### Usage

```r
data("trypanosome")
```

### Format

A data frame with 426 observations on the following 2 variables.

- `dead`: A logical vector.
- `dose`: A numeric vector.

### Details

The experimental technique involved inspection under the microscope of a representative aliquot of a suspension, all organisms appearing within two fields of view being classified either alive or dead. Hence the total numbers of organisms present at each dose and the number showing the quantal response were both random variables.

### Source


### References


### Examples

```r
data("trypanosome", package = "flexmix")
trypMix <- initFlexmix(cbind(Dead, 1-Dead) ~ 1, k = 2,
                       nrep = 5, data = trypanosome,
                       model = FLXRglmfix(family = "binomial",
                                           fixed = -log(Dose)))
```
whiskey

Survey Data on Brands of Scotch whiskey Consumed

**Description**

The data set is from Simmons Study of Media and Markets and contains the incidence matrix for scotch brands used in last year for those households who report consuming scotch.

**Usage**

data("whiskey")

**Format**

A data frame whiskey with 484 observations on the following 2 variables.

- **Freq** a numeric vector
- **Incidence** a matrix with 21 columns

Additional information on the brands is contained in the data frame whiskey_brands which is simultaneously loaded. This data frame contains 21 observations on the following 3 variables.

- **Brand** a character vector
- **Type** a factor with levels Blend Single Malt
- **Bottled** a factor with levels Domestic Foreign

**Details**

The dataset is taken from the bayesm package.

**Source**


**References**

Index

*Topic classes
  flexmix-class, 17
  FLXcomponent-class, 18
  FLXcontrol-class, 19
  FLXdist-class, 21
  FLXM-class, 23
  FLXnested-class, 42
  FLXP-class, 43

*Topic cluster
  flexmix, 14
  FLXfit, 22
  FLXMCmvbinary, 25
  FLXMCmvcombi, 26
  FLXMCmvnorm, 27
  FLXMCmvpois, 29
  FLXMRglmnet, 33
  FLXMRmgcv, 37
  plotEll, 53
  stepFlexmix, 61

*Topic datasets
  betablocker, 3
  bioChemists, 4
  BregFix, 7
  candy, 7
  dmf, 8
  ExLinear, 10
  ExNclus, 12
  ExNPreg, 12
  fabricfault, 13
  Mehta, 49
  NregFix, 50
  patent, 50
  salmonellataA98, 59
  seizure, 60
  tribolum, 64
  trypanosome, 65
  whiskey, 66

*Topic distribution
  rflexmix, 58

*Topic hplot
  plot-methods, 51

*Topic methods
  AIC-methods, 3
  BIC-methods, 4
  boot, 5
  EIC, 9
  fitted-methods, 14
  group, 44
  ICL, 45
  KLDiv, 46
  Lapply-methods, 47
  logLik-methods, 48
  plot-methods, 51
  posterior, 54
  refit-methods, 54
  relabel, 57

*Topic models
  FLXMCfactanal, 24
  FLXMRcondlogit, 30
  FLXMRglm, 31
  FLXMRglmfix, 32
  FLXMRlm, 35
  FLXMRlmnc, 36
  FLXMRmultinom, 39
  FLXMRrobglm, 40
  FLXMRziglm, 41
  FLXP, 43

*Topic regression
  flexmix, 14
  FLXfit, 22
  FLXMRcondlogit, 30
  FLXMRglm, 31
  FLXMRglmfix, 32
  FLXMRlmnc, 36
  FLXMRmultinom, 39
  FLXMRmgcv, 37
  FLXMRmultinom, 39
  stepFlexmix, 61

*Topic utilities

67
INDEX

FLXdist, 20
AIC, flexmix-method (AIC-methods), 3
AIC, stepFlexmix-method (AIC-methods), 3
AIC-methods, 3
betablocker, 3
BIC, flexmix-method (BIC-methods), 4
BIC, stepFlexmix-method (BIC-methods), 4
BIC-methods, 4
bioChemists, 4
boot, 5
boot, flexmix-method (boot), 5
BregFix, 7
candy, 7
clusters, flexmix, missing-method (posterior), 54
clusters, FLXboot, listOrData.frame-method (boot), 5
clusters, FLXdist, ANY-method (posterior), 54
coerce, list, FLXcontrol-method (FLXcontrol-class), 19
coerce, list, FLXnested-method (FLXnested-class), 42
coerce, NULL, FLXcontrol-method (FLXcontrol-class), 19
coerce, NULL, FLXnested-method (FLXnested-class), 42
coerce, numeric, FLXnested-method (FLXnested-class), 42
cv.glmer, 34
dmft, 8
EIC, 9
EIC, flexmix-method (EIC), 9
EIC, stepFlexmix-method (EIC), 9
eqscplot, 53
exLinear, 10
ExNclus, 12
ExNReg, 12
fabricfault, 13
fitted, flexmix-method (fitted-methods), 14
fitted, FLXMrGLM-method (fitted-methods), 14
fitted-methods, 14
flexmix, 14, 17, 18, 22–32, 34, 38, 39, 61, 62
flexmix, formula, ANY, ANY, ANY, FLXM-method (flexmix), 14
flexmix, formula, ANY, ANY, ANY, list-method (flexmix), 14
flexmix, formula, ANY, ANY, ANY, missing-method (flexmix), 14
flexmix-class, 17
FLXbclust (FLXMCmvbinary), 25
FLXboot-class (boot), 5
FLXcomponent-class, 18
FLXconstant (FLXP), 43
FLXcontrol-class, 19
FLXdeterminePostunscaled, FLXMRlmer-method (FLXMRlmer), 35
FLXdeterminePostunscaled, FLXMRlmm-method (FLXMRlmer), 35
FLXdistr, 20
FLXdistr-class, 21
FLXfit, 22
FLXfit, list-method (FLXfit), 22
FLXgetModelmatrix, FLXMRlmer-method (FLXMRlmer), 35
FLXgetModelmatrix, FLXMRlmm-method (FLXMRlmer), 35
FLXgetObs, FLXMRlmm-method (FLXMRlmer), 35
FLXglm (FLXMRglm), 31
FLXglmFix (FLXMRglmfix), 32
FLXgradlogLikfun, FLXMRziglm-method (FLXMRziglm), 41
FLXM-class, 23
FLXMC-class (FLXM-class), 23
FLXMCfactanal, 24
FLXMcclus (FLXMCmvnorm), 27
FLXMCmvbinary, 25, 26, 27
FLXMCmvcombi, 26
FLXMCmvnorm, 15, 23, 26, 27, 27, 53
FLXMCmvpois, 29
FLXMCnorm1 (FLXMCmvnorm), 27
FLXMR-class (FLXM-class), 23
FLXMRcondlogit, 30, 39
FLXMRglm, 15, 23, 31, 34, 38
FLXMRglmfix, 32
FLXMRglmnet, 33
posterior, 54
posterior, flexmix, missing-method
  (posterior), 54
posterior, FLXboot, listOrdata.frame-method
  (boot), 5
posterior, FLXdist, listOrdata.frame-method
  (posterior), 54
prcomp, 53
predict, FLXboot-method (boot), 5
predict, FLXdist-method (FLXdist-class), 21
predict, FLXMs-method (FLXdist-class), 21
predict, FLXMrGlm-method
  (FLXdist-class), 21
predict, FLXMrLm-method (FLXMrLmmc), 36
predict, FLXMrLmm-method (FLXMrLmer), 35
predict, FLXMrMgcv-method
  (FLXdist-class), 21
prior (FLXdist-class), 21
prior, flexmix-method (flexmix), 14
prior, FLXdist-method (FLXdist-class), 21
refit, flexmix-method (refit-methods), 54
refit, FLXMrZiglm-method (FLXMrZiglm), 41
refit-methods, 54
relabel, 57
relabel, FLXdist, character-method
  (relabel), 57
relabel, FLXdist, integer-method
  (relabel), 57
relabel, FLXdist, missing-method
  (relabel), 57
rflexmix, 58
rflexmix, flexmix, missing-method
  (rflexmix), 58
rflexmix, FLXdist, listOrdata.frame-method
  (rflexmix), 58
rflexmix, FLXdist, numeric-method
  (rflexmix), 58
rFLXM (rflexmix), 58
rFLXM, FLX, FLXcomponent-method
  (rflexmix), 58
rFLXM, FLX, list-method (rflexmix), 58
rFLXM, FLXMC, FLXcomponent-method
  (rflexmix), 58
rFLXM, FLXMCbinom, FLXcomponent-method
  (rflexmix), 58
rFLXM, FLXMCfactanal, FLXcomponent-method
  (FLXMCfactanal), 24
rFLXM, FLXMCmultinom, FLXcomponent-method
  (rflexmix), 58
rFLXM, FLXMRglm, FLXcomponent-method
  (rflexmix), 58
rFLXM, FLXMRglm, list-method (rflexmix), 58
rFLXM, FLXMRglmfix, list-method
  (rflexmix), 58
rFLXM, FLXMRlmm, FLXcomponent-method
  (FLXMRlmer), 35
rFLXM, FLXMRlmm, list-method (FLXMRlmer), 35
salmonellaTA98, 59
seizure, 60
show, Coefmat-method (refit-methods), 54
show, flexmix-method (flexmix), 14
show, FLXboot-method (boot), 5
show, FLXcomponent-method
  (FLXcomponent-class), 18
show, FLXdist-method (FLXdist), 20
show, FLXMs-method (FLXMs-class), 23
show, FLXP-method (FLXP), 43
show, FLXR-method (refit-methods), 54
show, stepFlexmix-method (stepFlexmix), 61
show, summary.flexmix-method (flexmix), 14
simulate, FLXdist-method (FLXdist), 20
stepFlexmix, 19, 61
stepFlexmix-class (stepFlexmix), 61
summary, flexmix-method (flexmix), 14
summary, FLXRmstep-method
  (refit-methods), 54
summary, FLXRroptim-method
  (refit-methods), 54
tribolium, 64
trypanosome, 65
unique, stepFlexmix-method
  (stepFlexmix), 61
update.formula, 24–27, 29–31, 34, 35,
  37–41
whiskey, 66
whiskey_brands (whiskey), 66
xyplot, 55