

# Package ‘flexmix’

May 29, 2009

**Version** 2.2-3

**Date** 2009-05-28

**Author** Friedrich Leisch and Bettina Gruen

**Maintainer** Bettina Gruen <Bettina.Gruen@wu.ac.at>

**Title** Flexible Mixture Modeling

**Depends** lattice, modeltools (>= 0.2-16), multcomp

**Imports** methods, stats, stats4

**Suggests** MASS, ellipse, grid, mvtnorm, nnet

**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.

**License** GPL-2

**URL** <http://www.aasc.or.at/mixtures>

**LazyLoad** yes

**Repository** CRAN

**Date/Publication** 2009-05-29 11:43:31

## R topics documented:

AIC-methods . . . . .	3
betablocker . . . . .	3
BIC-methods . . . . .	4
bioChemists . . . . .	4
BregFix . . . . .	5
candy . . . . .	6
dmft . . . . .	6

ExLinear	7
ExNclus	9
ExNPreg	10
fabricfault	10
fitted-methods	11
flexmix	12
flexmix-class	14
FLXcomponent-class	15
FLXcontrol-class	16
FLXdist	17
FLXdist-class	18
FLXfit	19
FLXM-class	20
FLXMCmvbinary	21
FLXMCmvcombi	22
FLXMCmvnorm	23
FLXMCmvpois	24
FLXMRglm	25
FLXMRglmfix	26
FLXMRrobglm	27
FLXMRziglm	29
FLXnested-class	30
FLXP	30
FLXP-class	31
group	31
ICL	32
KLdiv	33
Lapply-methods	35
logLik-methods	36
Mehta	36
NregFix	37
patent	38
plot-methods	39
plotEll	40
posterior	41
refit-methods	42
relabel	44
rflexmix	45
salmonellaTA98	46
seizure	47
stepFlexmix	48
tribolium	50
trypanosome	51
whiskey	52

---

AIC-methods

*Methods for Function AIC*

---

### 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

G. McLachlan and D. Peel (2000): Finite Mixture Models. John Wiley and Sons Inc. <http://www.maths.uq.edu.au/~gjm/DATA/mmdata.html>

### References

M. Aitkin (1999): Meta-analysis by random effect modelling in generalized linear models. *Statistics in medicine* 18, pages 2343-2351.

S. Yusuf, R. Peto, J. Lewis, R. Collins and P. Sleight (1985): Beta blockade during and after myocardial infarction: an overview of the randomized trials. *Progress in Cardiovascular Diseases* 27, pages 335-371.

**Examples**

```
data("betablocker")
betaMix <- stepFlexmix(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

*Article production 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>

**References**

Long, J. Scott (1990). "The origins of sex difference in science". *Social Forces*. 68:1297-1315.

Long, J. Scott (1997). *Regression Models for Categorical and Limited Dependent Variables*. Thousand Oaks, California: Sage.

---

BregFix

*Artificial Example for Binomial Regression*

---

**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")
Model <- FLXMRglmfix(family="binomial",
                    nested = list(formula = c(~x, ~0), k = c(2,1)))
Conc <- FLXPmultinom(~w)
FittedBin <- stepFlexmix(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")
```

### Format

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

**Packages** a numeric vector

**Freq** a numeric vector

### Source

D. Boehning, E. Dietz, P. Schlattmann (1998): Recent Developments in Computer-Assisted Analysis of Mixtures. *Biometrics* 54(2), 525–536.

### References

J. Magidson, J. K. Vermunt (2004): Latent Class Models. In D. W. Kaplan (ed.), *The Sage Handbook of Quantitative Methodology for the Social Sciences*, 175–198. Thousand Oakes: Sage Publications.

D. Boehning, E. Dietz, P. Schlattmann (1998): Recent Developments in Computer-Assisted Analysis of Mixtures. *Biometrics* 54(2), 525–536.

W. R. Dillon, A. Kumar (1994): Latent structure and other mixture models in marketing: An integrative survey and overview. In R. P. Bagozzi (ed.), *Advanced methods of marketing research*, 352–388. Cambridge, UK: Blackwell.

---

 dmft

*Dental data*


---

### 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**

D. Boehning, E. Dietz, P. Schlattmann, L. Mendonca and U. Kirchner (1999): The zero-inflated Poisson model and the decayed, missing and filled teeth index in dental epidemiology. *Journal of the Royal Statistical Society A* 162(2), pages 195-209.

**Examples**

```
data("dmft")
dmft.flx <- stepFlexmix(End ~ 1, data = dmft, k = 2,
                      model = FLXMRglmfix(family = "poisson",
                      fixed = ~ Gender + Ethnic + Treatment))
```

**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**

<code>beta</code>	A matrix of regression coefficients. Each row corresponds to a variable, each column to a mixture component. The first row is used as intercept.
<code>n</code>	Integer, the number of observations per component.
<code>xdist</code>	Character, name of a random number function for the explanatory variables.
<code>xdist.args</code>	List, arguments for the random number functions.
<code>family</code>	A character string naming a GLM family. Only "gaussian" and "poisson" are implemented at the moment.
<code>sd</code>	Numeric, the error standard deviation for each component for Gaussian responses.
<code>...</code>	Used as default for <code>xdist.args</code> 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)-1`. 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**

```
## 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, "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, "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")
require("MASS")
eqsplot(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 `ExNPreg(100)` using `set.seed(2602)`, it has been saved as a data set for simplicity of examples only.

### Examples

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

---

 fabricfault

*Fabric Faults*


---

### Description

Number of faults in rolls of a textile fabric.

### Usage

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

### Format

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

**Length** Length of role (m).

**Faults** Number of faults.

**Source**

G. McLachlan and D. Peel (2000): Finite Mixture Models. John Wiley and Sons Inc. <http://www.maths.uq.edu.au/~gjm/DATA/mmdata.html>

**References**

A. F. Bissell (1972): A Negative Binomial Model with Varying Element Sizes *Biometrika* 59, pages 435-441.

M. Aitkin (1996): A general maximum likelihood analysis of overdispersion in generalized linear models. *Statistics and Computing* 6, pages 251-262.

**Examples**

```
data("fabricfault")
fabricMix <- stepFlexmix(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**

```
## 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

```
data("NPreg")
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

```
flexmix(formula, data = list(), k = NULL, cluster = NULL,
        model=NULL, concomitant=NULL, control = NULL,
        weights = NULL)
## S4 method for signature 'flexmix':
summary(object, eps=1e-4, ...)
```

## Arguments

<code>formula</code>	A symbolic description of the model to be fit. The general form is $y \sim x   g$ where $y$ is the response, $x$ the set of predictors and $g$ an optional grouping factor for repeated measurements.
<code>data</code>	An optional data frame containing the variables in the model.
<code>k</code>	Number of clusters (not needed if <code>cluster</code> is specified).
<code>cluster</code>	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.
<code>weights</code>	An optional vector of weights to be used in the fitting process. Should be 'NULL', a numeric vector or a formula.
<code>model</code>	Object of class <code>FLXM</code> or list of <code>FLXM</code> objects. Default is the object returned by calling <code>FLXMRglm()</code> .
<code>concomitant</code>	Object of class <code>FLXP</code> . Default is the object returned by calling <code>FLXPconstant</code> .
<code>control</code>	Object of class <code>FLXcontrol</code> or a named list.
<code>object</code>	Object of class <code>flexmix</code> .
<code>eps</code>	Probabilities below this threshold are treated as zero in the summary method.
<code>...</code>	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

Friedrich Leisch. FlexMix: A general framework for finite mixture models and latent class regression in R. *Journal of Statistical Software*, 11(8), 2004. <http://www.jstatsoft.org/v11/i08/>

## See Also

[plot-methods](#)

## Examples

```
data("NPreg")

## 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:
ex4 <- flexmix(yp~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 <- stepFlexmix(cbind(yb,1-yb)~x, data=NPreg, k=2,
                  model=FLXMRglm(family="binomial"), nrep=5)
table(NPreg$class, clusters(ex5))

ex6 <- stepFlexmix(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 FLXM 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

---

FLXcontrol-class    *Class "FLXcontrol"*

---

### 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

```
## 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**

<code>formula</code>	A symbolic description of the model to be fit. The general form is $\sim x   g$ where $x$ is the set of predictors and $g$ an optional grouping factor for repeated measurements.
<code>k</code>	Integer specifying the number of cluster or a numeric vector of length equal to the length of components, specifying the prior probabilities of clusters.
<code>model</code>	Object of class <code>FLXM</code> or a list of <code>FLXM</code> objects. Default is the object returned by calling <code>FLXMRglm()</code> .
<code>components</code>	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.
<code>concomitant</code>	Object of class <code>FLXconcomitant</code> 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

```
## 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

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

FLXdists

---

FLXfit

*Fitter Function for FlexMix Models*

---

### Description

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

### Usage

```
FLXfit(model, concomitant, control, postunscaled = NULL, groups,
        weights)
```

### Arguments

<code>model</code>	List of FLXM objects.
<code>concomitant</code>	Object of class FLXP.
<code>control</code>	Object of class FLXcontrol.
<code>weights</code>	A numeric vector of weights to be used in the fitting process.
<code>postunscaled</code>	Initial a-posteriori probabilities of the observations at the start of the EM algorithm.
<code>groups</code>	List with components <code>group</code> which is a factor with optional grouping of observations and <code>groupfirst</code> 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

**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 [FLXMCmvnorm](#).

**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

---

FLXMCmvbinary	<i>FlexMix Binary Clustering Driver</i>
---------------	---

---

### Description

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

### Usage

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

### Arguments

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

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

### Author(s)

Friedrich Leisch and Bettina Gruen

### See Also

`flexmix`

---

`FLXMCmvcombi`*FlexMix Binary and Gaussian Clustering Driver*

---

### 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 `FLXMCmvbinary` 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

### See Also

`flexmix`, `FLXMCmvbinary`, `FLXMCmvnorm`

### 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)))
```

---

FLXMCmvnorm

*FlexMix Clustering Demo Driver*

---

## Description

This is a demo driver for `flexmix` implementing model-based clustering of Gaussian data.

## Usage

```
FLXMCmvnorm(formula = . ~ ., diagonal = TRUE)
```

## Arguments

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

## Value

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

## Author(s)

Friedrich Leisch and Bettina Gruen

## References

Friedrich Leisch. FlexMix: A general framework for finite mixture models and latent class regression in R. *Journal of Statistical Software*, 11(8), 2004. <http://www.jstatsoft.org/v11/i08/>

**See Also**[flexmix](#)**Examples**

```

data("Nclus")

require("MASS")
eqsplot(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=FLXMCmvnorm(diag=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,]
p

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

```

---

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`

---

 FLXMRglm

*FlexMix Interface to Generalized Linear Models*


---

**Description**

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

**Usage**

```
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 regression in R. *Journal of Statistical Software*, 11(8), 2004. <http://www.jstatsoft.org/v11/i08/>

**See Also**

`flexmix`, `glm`

---

FLXMRglmfix

*FlexMix Interface to Generalized Linear Models with fixed coefficients*

---

**Description**

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

<code>formula</code>	A formula which is interpreted relative to the formula specified in the call to <code>flexmix</code> using <code>update.formula</code> . Default is to use the original <code>flexmix</code> model formula.
<code>fixed</code>	A formula which specifies the additional regressors for the fixed (constant) coefficients.
<code>varFix</code>	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.
<code>nested</code>	An object of class <code>FLXnested</code> or a list specifying the nested structure.
<code>family</code>	A character string naming a <code>glm</code> family function.
<code>offset</code>	This can be used to specify an <i>a priori</i> 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**

```
data("NPreg")
ex <- flexmix(yn~x|id2, data=NPreg, k=2,
             model=FLXMRglm(yn~.+I(x^2)))
ex.fix <- flexmix(yn~x|id2, data = NPreg,
                model=FLXMRglmfix(nested = list(k = c(1,1),
                                                formula = c(~0, ~I(x^2))))
summary(refit(ex))
summary(refit(ex.fix))
```

---

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**

<code>formula</code>	A formula which is interpreted relative to the formula specified in the call to <code>flexmix</code> using <code>update.formula</code> . Default is to use the original <code>flexmix</code> model formula.
<code>family</code>	A character string naming a <code>glm</code> family function.
<code>bgw</code>	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 ( <code>bgw=TRUE</code> ).
<code>...</code>	passed to <code>FLXMRglm</code>

**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**

Friedrich Leisch. Modelling background noise in finite mixtures of generalized linear regression models. In Paula Brito, editor, *Compstat 2008-Proceedings in Computational Statistics*, pages 385-396. Physica Verlag, Heidelberg, Germany, 2008. Preprint available at <http://epub.ub.uni-muenchen.de/6332/>.

**Examples**

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

## Estimation without (f2) and with (f3) background component
f2 <- flexmix(yn~x+I(x^2), data=DATA2, k=2)
f3 <- flexmix(yn~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(yn~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(yn~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)
```

**Description**

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

**Usage**

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

**Arguments**

formula	A formula which is interpreted relative to the formula specified in the call to flexmix using <code>update.formula</code> . Default is to use the original flexmix model formula.
family	A character string naming a <code>glm</code> family function.
...	passed to FLXMRglm

**Value**

Returns an object of class FLXMRziglm.

**Note**

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

**Author(s)**

Friedrich Leisch and Bettina Gruen

**Examples**

```
data("dmft")
Model <- FLXMRziglm(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                      *Creates the concomitant variable model.*

---

### 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**

```
## 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**

```
## 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**

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

**Examples**

```
data("NPrep")
ex1 <- flexmix(yn~x+I(x^2), data=NPrep, k=2)
ICL(ex1)
```

---

KLdiv

*Kullback-Leibler Divergence*


---

**Description**

Estimate the Kullback-Leibler divergence of several distributions.

**Usage**

```
## 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 `eps` is given to each observation point for the modified densities.

## Author(s)

Friedrich Leisch and Bettina Gruen

## References

S. Kullback and R. A. Leibler. On information and sufficiency. *The Annals of Mathematical Statistics* 22(1), pages 79-86, 1951.

Friedrich Leisch. Exploring the structure of mixture model components. In Jaromir Antoch, editor, *Compstat 2004 - Proceedings in Computational Statistics*, pages 1405-1412. Physika Verlag, Heidelberg, Germany, 2004. ISBN 3-7908-1554-3.

## Examples

```
## 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")
KLdiv(y)

if (require("mlbench")) {
  set.seed(2606)
  x <- mlbench.smiley()$x
  modell <- flexmix(x~1, k=9, model=FLXmclust(diag=FALSE),
                  control = list(minprior=0))
  plotE11(modell, x)
  KLdiv(modell)
}
```

**Description**

Apply a function to each component of a finite mixture

**Usage**

```
## S4 method for signature 'FLXRMstep':
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 backquoted 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")
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

---

`Mehta`*Mehta Trial*

---

**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**

M. Aitkin (1999): Meta-analysis by random effect modelling in generalized linear models. *Statistics in medicine* 18, pages 2343-2351.

**References**

C.R. Mehta, N.R. Patel and P. Senchaudhuri (1988): Importance sampling for estimating exact probabilities in permutational inference. *Journal of the American Statistical Association* 83, pages 999-1005.

**Examples**

```
data("Mehta")
mehtaMix <- stepFlexmix(cbind(Response, Total-Response) ~ 1|Site,
                        data=Mehta, nrep=5, k=3,
                        model=FLXMRglmfix(family="binomial",
                                           fixed=~ Drug),
                        control=list(minprior=0.04))
```

---

NregFix

*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**

```
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**

```
data("NregFix")
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 <- stepFlexmix(y ~ 1, model = Model, data = NregFix, k = 3,
                           concomitant = FLXPmultinom(~ w), nrep=5)
fittedModel
summary(refit(fittedModel))
```

patent

*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**

```
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**

P. Wang, I.M. Cockburn and M.L. Puterman (1998): Analysis of Patent Data - A Mixed-Poisson-Regression-Model Approach. *Journal of Business & Economic Statistics* 16 (1), pages 27-41.

**References**

B.H. Hall, C. Cummins, E. Laderman and J. Mundy (1988): The R&D Master File Documentation. Technical Working Paper 72, National Bureau of Economic Research. Cambridge, MA.

**Examples**

```
data("patent")
patentMix <- stepFlexmix(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**

```
## 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**

<code>x</code>	An object of class "flexmix".
<code>y</code>	Not used.
<code>mark</code>	Integer: mark posteriors of this component.
<code>markcol</code>	Color used for marking components.
<code>col</code>	Color used for the bars.
<code>eps</code>	Posteriors smaller than <code>eps</code> are ignored.
<code>root</code>	If <code>TRUE</code> , a rootogram of the posterior probabilities is drawn, otherwise a standard histogram.
<code>ylim</code>	A logical value or a numeric vector of length 2. If <code>TRUE</code> , the y axes of all rootograms are aligned to have the same limits, if <code>FALSE</code> each y axis is scaled separately. If a numeric vector is specified it is used as usual.
<code>main</code>	Main title of the plot.
<code>xlab</code>	Label of x-axis.
<code>ylab</code>	Label of y-axis.
<code>as.table</code>	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.
<code>endpoints</code>	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.
<code>...</code>	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 separated 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

Friedrich Leisch. FlexMix: A general framework for finite mixture models and latent class regression in R. *Journal of Statistical Software*, 11(8), 2004. <http://www.jstatsoft.org/v11/i08/>

Jeremy Tantrum, Alejandro Murua and Werner Stuetzle. Assessment and pruning of hierarchical model based clustering. *Proceedings of the 9th ACM SIGKDD international conference on Knowledge Discovery and Data Mining*, pages 197-205. ACM Press, New York, NY, USA, 2003.

Friedrich Leisch. Exploring the structure of mixture model components. In Jaromir Antoch, editor, *Compstat 2004 - Proceedings in Computational Statistics*, pages 1405-1412. Physika Verlag, Heidelberg, Germany, 2004. ISBN 3-7908-1554-3.

---

plotEll

*Plot Confidence Ellipses for FLXMCmvnorm Results*

---

## Description

Plot confidence ellipses for mixtures of Gaussians fitted using `FLXMCmvnorm`.

## Usage

```
plotEll(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 <code>flexmix</code> with a fitted <code>FLXMCmvnorm</code> 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 <code>prcomp</code> is supported.
points	Logical, shall data points be plotted?
eqscale	Logical, plot using <code>eqsplot</code> ?
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**

```
## 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**

```
## 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 <code>optim</code> 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.

<code>which</code>	Specifies if a component specific model or the concomitant variable model is used.
<code>x</code>	An object of class "FLXROptim"
<code>y</code>	Missing object.
<code>bycluster</code>	A logical if the parameters should be group by cluster or by variable.
<code>alpha</code>	Numeric indicating the significance level.
<code>components</code>	Numeric vector specifying which components are plotted. The default is to plot all components.
<code>labels</code>	Character vector specifying the variable names used.
<code>significance</code>	A logical indicating if non-significant coefficients are shaded in a lighter grey.
<code>xlab</code>	String for the x-axis label.
<code>ylab</code>	String for the y-axis label.
<code>ci</code>	A logical indicating if significant and insignificant parameter estimates are shaded differently.
<code>scales</code>	See argument of the same name for function <code>xyplot</code> .
<code>as.table</code>	See arguments of the same name for function <code>xyplot</code> .
<code>horizontal</code>	See arguments of the same name for function <code>xyplot</code> .
<code>...</code>	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 from 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 for refitting `FLXMRglm` components using `method = "mstep"`. 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

Friedrich Leisch. FlexMix: A general framework for finite mixture models and latent class regression in R. *Journal of Statistical Software*, 11(8), 2004. <http://www.jstatsoft.org/v11/i08/>

### Examples

```
data("NPreg")
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, and then one EM step is made initialized with the posterior probabilities of the previous fit (because of this single EM step estimates may change slightly even if EM converged previously).

### Usage

```
relabel(object, by, ...)
## S4 method for signature 'flexmix, character':
relabel(object, by, data, ...)
```

### Arguments

<code>object</code>	An object of class "flexmix".
<code>by</code>	Name (or unique substring) of a parameter.
<code>data</code>	Data frame or list containing the data. If missing, an attempt is made to recover the data from the call, this typically works only when used at the prompt and no model contains extra variables in the model-specific formula .
<code>...</code>	Currently not used.

**Author(s)**

Friedrich Leisch and Bettina Gruen

**Examples**

```

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:
r1 <- relabel(f1, by="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 <code>flexmix</code> or an unfitted of class <code>FLXdist</code> .
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

<code>y</code>	Random sample
<code>group</code>	Grouping factor
<code>class</code>	Class membership

## Author(s)

Bettina Gruen

## Examples

```
example(flexmix)
sample <- rflexmix(ex1)
```

---

salmonellaTA98	<i>Salmonella reverse mutagenicity assay</i>
----------------	--

---

## 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 testes at each of six dose levels of quinolinediameter 4.5 feet of the ground, inches.

## Details

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

**Source**

Margolin, B.J., Kaplan, N. and Zeiger, E. (1981) Statistical analysis of the Ames Salmonella/microsome test, *Proc. Natl. Acad. Sci. USA*, **76**, 3779–3783.

**References**

Breslow, N.E. (1984), Extra-Poisson variation in log-linear models, *Applied Statistics*, **33**, 38–44.

Wang, P., Puterman, M.L., Cockburn, I.M., and Le, N.D. (1996) Mixed Poisson regression models with covariate dependent rates, *Biometrics*, **52**, 381–400.

**Examples**

```
data("salmonellaTA98")
salmonMix <- stepFlexmix(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)
```

---

 seizure

*Epileptic Seizure Data*


---

**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**

P. Wang, M. Puterman, I. Cockburn, and N. Le (1996): Mixed poisson regression models with covariate dependent rates. *Biometrics* 52, pages 381-400.

**References**

B. Gruen and F. Leisch (2004): Bootstrapping finite mixture models. In J. Antoch, editor, *Compstat 2004 - Proceedings in Computational Statistics*, pages 1115-1122. Physika Verlag, Heidelberg, Germany, ISBN 3-7908-1554-3.

**Examples**

```
data("seizure")
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

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**

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

## S4 method for signature 'stepFlexmix, missing':
```

```

plot(x, y, what=c("AIC", "BIC", "ICL"),
     xlab=NULL, ylab=NULL, legend="topright", ...)

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

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

```

## Arguments

<code>...</code>	Passed to <code>flexmix</code> (or <code>matplot</code> in the <code>plot</code> method).
<code>k</code>	A vector of integers passed in turn to the <code>k</code> argument of <code>flexmix</code> .
<code>nrep</code>	For each value of <code>k</code> run <code>flexmix</code> <code>nrep</code> times and keep only the solution with maximum likelihood.
<code>verbose</code>	If TRUE, show progress information during computations.
<code>drop</code>	If TRUE and <code>k</code> is of length 1, then a single <code>flexmix</code> object is returned instead of a "stepFlexmix" object.
<code>unique</code>	If TRUE, then <code>unique()</code> is called on the result, see below.
<code>x, object</code>	An object of class "stepFlexmix".
<code>y</code>	Not used.
<code>what</code>	Character vector naming information criteria to plot. Functions of the same name must exist, which take a <code>stepFlexmix</code> object as input and return a numeric vector like <code>AIC</code> , <code>stepFlexmix-method</code> (see examples below).
<code>xlab, ylab</code>	Graphical parameters.
<code>legend</code>	If not FALSE and <code>what</code> contains more than 1 element, a legend is placed at the specified location, see <a href="#">legend</a> for details.
<code>which</code>	Number of model to get. If character, interpreted as number of components or name of an information criterion.
<code>incomparables</code>	A vector of values that cannot be compared. Currently, '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 `length(k) > 1`, else directly an object of class "flexmix".

If `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 `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 `unique()` on objects of class "stepFlexmix".

**Author(s)**

Friedrich Leisch and Bettina Gruen

**References**

Friedrich Leisch. FlexMix: A general framework for finite mixture models and latent class regression kin R. *Journal of Statistical Software*, 11(8), 2004. <http://www.jstatsoft.org/v11/i08/>

**Examples**

```

data("Nclus")
set.seed(511)

## try 5 times for k=4
ex1 <- stepFlexmix(Nclus~1, k=4, model=FLXMCmvnorm(diag=FALSE), nrep=5)
ex1

## now 3 times each for k=2:6, specify control parameter
ex2 <- stepFlexmix(Nclus~1, k=2:6, model=FLXMCmvnorm(diag=FALSE),
                  control=list(minprior=0), nrep=3)
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=2:6, with larger minimum prior
ex3 <- stepFlexmix(Nclus~1, k=2:6, model=FLXMCmvnorm(diag=FALSE),
                  control=list(minprior=0.1), nrep=1)
ex3

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

```

---

tribolium

*Tribolium beetles*

---

**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**

```
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**

P. Wang and M.L. Puterman (1998): Mixed Logistic Regression Models. *Journal of Agricultural, Biological, and Environmental Statistics* 3 (2), pages 175-200,

**Examples**

```
data("tribolium")
tribMix <- stepFlexmix(cbind(Remaining, Total - Remaining) ~ Species,
                      k = 2, nrep=5, data = tribolium,
                      model = FLXMRglm(family = "binomial"))
```

---

trypanosome

*Trypanosome*


---

**Description**

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**

```
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**

R. Ashford and P.J. Walker (1972): Quantal Response Analysis for a Mixture of Populations. *Biometrics* 28, pages 981-988.

**References**

D.A. Follmann and D. Lambert (1989): Generalizing Logistic Regression by Nonparametric Mixing. *Journal of the American Statistical Association* 84(405), pages 195-300.

**Examples**

```
data("trypanosome")
trypMix <- stepFlexmix(cbind(Dead, 1-Dead) ~ 1, k = 2,
                      nrep=5, data = trypanosome,
                      model = FLXMRglmfix(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**

Peter Rossi and Rob McCulloch. (2006). `bayesm`: Bayesian Inference for Marketing/Micro-econometrics. R package version 2.0-8. <http://gsbwww.uchicago.edu/fac/peter.rossi/research/bsm.html>

**References**

Edwards, Y. and G. Allenby (2003), "Multivariate Analysis of Multiple Response Data," *JMR* 40, 321-334.

# Index

## \*Topic **classes**

- flexmix-class, 13
- FLXcomponent-class, 14
- FLXcontrol-class, 15
- FLXdists-class, 17
- FLXM-class, 19
- FLXnested-class, 29
- FLXP-class, 30

## \*Topic **cluster**

- flexmix, 11
- FLXfit, 18
- FLXMCmvbinary, 20
- FLXMCmvcombi, 21
- FLXMCmvnorm, 22
- FLXMCmvpois, 23
- plotEll, 39
- stepFlexmix, 47

## \*Topic **datasets**

- betablocker, 2
- bioChemists, 3
- BregFix, 4
- candy, 5
- dmft, 5
- ExLinear, 6
- ExNclus, 8
- ExNPreg, 9
- fabricfault, 9
- Mehta, 35
- NregFix, 36
- patent, 37
- salmonellaTA98, 45
- seizure, 46
- tribolium, 49
- trypanosome, 50
- whiskey, 51

## \*Topic **distribution**

- rflexmix, 44

## \*Topic **hplot**

- plot-methods, 38

## \*Topic **methods**

- AIC-methods, 2
- BIC-methods, 3
- fitted-methods, 10
- group, 30
- ICL, 31
- KLdiv, 32
- Lapply-methods, 34
- logLik-methods, 35
- plot-methods, 38
- posterior, 40
- refit-methods, 41
- relabel, 43

## \*Topic **models**

- FLXMRglm, 24
- FLXMRglmfix, 25
- FLXMRrobglm, 26
- FLXMRziglm, 28
- FLXP, 29

## \*Topic **regression**

- flexmix, 11
- FLXfit, 18
- FLXMRglm, 24
- FLXMRglmfix, 25
- stepFlexmix, 47

## \*Topic **utilities**

- FLXdists, 16

AIC, flexmix-method (*AIC-methods*),  
2

AIC, stepFlexmix-method  
(*AIC-methods*), 2

AIC-methods, 2

betablocker, 2

BIC, flexmix-method (*BIC-methods*),  
3

BIC, stepFlexmix-method  
(*BIC-methods*), 3

BIC-methods, 3

- bioChemists, 3
- BregFix, 4
- candy, 5
- clusters, flexmix, missing-method  
(*posterior*), 40
- clusters, FLXdists, ANY-method  
(*posterior*), 40
- coerce, list, FLXcontrol-method  
(*FLXcontrol-class*), 15
- coerce, list, FLXnested-method  
(*FLXnested-class*), 29
- coerce, NULL, FLXcontrol-method  
(*FLXcontrol-class*), 15
- coerce, NULL, FLXnested-method  
(*FLXnested-class*), 29
- coerce, numeric, FLXnested-method  
(*FLXnested-class*), 29
- dmft, 5
- eqscplot, 40
- ExLinear, 6
- ExNclus, 8
- ExNPreg, 9
- fabricfault, 9
- fitted, flexmix-method  
(*fitted-methods*), 10
- fitted, FLXM-method  
(*fitted-methods*), 10
- fitted, FLXR-method  
(*fitted-methods*), 10
- fitted, FLXMRglm-method  
(*fitted-methods*), 10
- fitted-methods, 10
- flexmix, 11, 13, 14, 18–25, 48
- flexmix, formula, ANY, ANY, ANY, FLXM-method  
(*flexmix*), 11
- flexmix, formula, ANY, ANY, ANY, list-method  
(*flexmix*), 11
- flexmix, formula, ANY, ANY, ANY, missing-method  
(*flexmix*), 11
- flexmix-class, 19, 38
- flexmix-class, 13
- FLXbclust (*FLXMCmvbinary*), 20
- FLXcomponent-class, 14
- FLXconstant (*FLXP*), 29
- FLXcontrol-class, 15
- FLXdists, 16
- FLXdists-class, 17
- FLXfit, 18
- FLXfit, list-method (*FLXfit*), 18
- FLXgetDesign, FLXMRziglm-method  
(*FLXMRziglm*), 28
- FLXglm (*FLXMRglm*), 24
- FLXglmFix (*FLXMRglmfix*), 25
- FLXgradlogLikfun, FLXMRziglm-method  
(*FLXMRziglm*), 28
- FLXM-class, 19
- FLXMC-class (*FLXM-class*), 19
- FLXmclust (*FLXMCmvnorm*), 22
- FLXMCmvbinary, 20, 21
- FLXMCmvcombi, 21
- FLXMCmvnorm, 12, 19, 21, 22, 39, 40
- FLXMCmvpois, 23
- FLXMR-class (*FLXM-class*), 19
- FLXMRglm, 11, 12, 19, 24
- FLXMRglmfix, 25
- FLXMRrobglm, 26
- FLXMRrobglm-class (*FLXMRrobglm*),  
26
- FLXMRziglm, 28
- FLXMRziglm-class (*FLXMRziglm*), 28
- FLXmultinom (*FLXP*), 29
- FLXnested-class, 29
- FLXP, 29
- FLXP-class, 30
- FLXPconstant, 11, 30
- FLXPconstant (*FLXP*), 29
- FLXPconstant-class (*FLXP-class*),  
30
- FLXPmultinom, 30
- FLXPmultinom (*FLXP*), 29
- FLXPmultinom-class (*FLXP-class*),  
30
- FLXreplaceParameters, FLXMRziglm-method  
(*FLXMRziglm*), 28
- FLXRmstep-class (*refit-methods*),  
41
- FLXroptim-class (*refit-methods*),  
41
- getModel, stepFlexmix-method  
(*stepFlexmix*), 47
- glm, 24–26, 28
- group, 30
- group, flexmix-method (*group*), 30

- group, FLXM-method (*group*), 30
- group, FLXMRglmfix-method (*group*), 30
- group-methods (*group*), 30
- ICL, 31
- ICL, flexmix-method (*ICL*), 31
- ICL, stepFlexmix-method (*ICL*), 31
- initialize, FLXnested-method (*FLXnested-class*), 29
- initialize, FLXP-method (*FLXP-class*), 30
- KLdiv, 32
- KLdiv, flexmix-method (*KLdiv*), 32
- KLdiv, FLXMC-method (*KLdiv*), 32
- KLdiv, FLXMRglm-method (*KLdiv*), 32
- KLdiv, matrix-method (*KLdiv*), 32
- Lapply, FLXRmstep-method (*Lapply-methods*), 34
- Lapply-methods, 34
- legend, 48
- logLik, flexmix-method (*logLik-methods*), 35
- logLik-methods, 35
- matplot, 48
- Mehta, 35
- Nclus (*ExNclus*), 8
- NPreg (*ExNPreg*), 9
- NregFix, 36
- optim, 41
- parameters, FLXdist-method (*FLXdist-class*), 17
- patent, 37
- plot, flexmix, missing-method (*plot-methods*), 38
- plot, FLXRoptim, missing-method (*refit-methods*), 41
- plot, stepFlexmix, missing-method (*stepFlexmix*), 47
- plot-methods, 12
- plot-methods, 38
- plotEll, 39
- posterior, 40
- posterior, flexmix, missing-method (*posterior*), 40
- posterior, FLXdist, listOrdata.frame-method (*posterior*), 40
- prcomp, 40
- predict, FLXdist-method (*FLXdist-class*), 17
- predict, FLXM-method (*FLXdist-class*), 17
- predict, FLXMRglm-method (*FLXdist-class*), 17
- prior (*FLXdist-class*), 17
- prior, FLXdist-method (*FLXdist-class*), 17
- refit, flexmix-method (*refit-methods*), 41
- refit, FLXMRzigm-method (*FLXMRzigm*), 28
- refit-methods, 41
- relabel, 43
- relabel, flexmix, character-method (*relabel*), 43
- rflexmix, 44
- rflexmix, flexmix, missing-method (*rflexmix*), 44
- rflexmix, FLXdist, listOrdata.frame-method (*rflexmix*), 44
- rflexmix, FLXdist, numeric-method (*rflexmix*), 44
- rFLXM (*rflexmix*), 44
- rFLXM, FLXM, FLXcomponent-method (*rflexmix*), 44
- rFLXM, FLXM, list-method (*rflexmix*), 44
- rFLXM, FLXMC, FLXcomponent-method (*rflexmix*), 44
- rFLXM, FLXMCbinom, FLXcomponent-method (*rflexmix*), 44
- rFLXM, FLXMCmultinom, FLXcomponent-method (*rflexmix*), 44
- rFLXM, FLXMRglm, FLXcomponent-method (*rflexmix*), 44
- rFLXM, FLXMRglm, list-method (*rflexmix*), 44
- rFLXM, FLXMRglmfix, list-method (*rflexmix*), 44
- salmonellaTA98, 45

seizure, [46](#)

show, Coefmat-method  
(*refit-methods*), [41](#)

show, flexmix-method (*flexmix*), [11](#)

show, FLXcomponent-method  
(*FLXcomponent-class*), [14](#)

show, FLXdists-method (*FLXdists*), [16](#)

show, FLXM-method (*FLXM-class*), [19](#)

show, FLXP-method (*FLXP*), [29](#)

show, FLXR-method (*refit-methods*),  
[41](#)

show, stepFlexmix-method  
(*stepFlexmix*), [47](#)

show, summary.flexmix-method  
(*flexmix*), [11](#)

simulate, FLXdists-method  
(*FLXdists*), [16](#)

stepFlexmix, [15](#), [47](#)

stepFlexmix-class (*stepFlexmix*),  
[47](#)

summary, flexmix-method (*flexmix*),  
[11](#)

summary, FLXRmstep-method  
(*refit-methods*), [41](#)

summary, FLXROptim-method  
(*refit-methods*), [41](#)

tribolium, [49](#)

trypanosome, [50](#)

unique, stepFlexmix-method  
(*stepFlexmix*), [47](#)

update.formula, [20–22](#), [24](#), [26](#), [28](#)

whiskey, [51](#)

whiskey\_brands (*whiskey*), [51](#)

xyplot, [42](#)