fmm — Finite mixture models using the fmm prefix

Description

The `fmm` prefix fits finite mixture models; see [FMM] **fmm estimation** for the list of supported commands.

Quick start

Mixture of three normal distributions of `y`

```
fmm 3: regress y
```

Mixture of three linear regression models of `y` on `x1` and `x2`

```
fmm 3: regress y x1 x2
```

Same as above, but with class probabilities depending on `z1` and `z2`

```
fmm 3, lcprob(z1 z2): regress y x1 x2
```

Same as above, but with additional class-specific regression covariates `x3`, `x4`, and `x5`

```
fmm, lcprob(z1 z2): (regress y x1 x2 x3)    ///
          (regress y x1 x2 x4)    ///
          (regress y x1 x2 x5)
```

Same as above, but with additional class-specific probability covariates `z3` and `z4`

```
fmm: (regress y x1 x2 x3)            ///
   (regress y x1 x2 x4, lcprob(z1 z2 z3))  ///
   (regress y x1 x2 x5, lcprob(z1 z2 z4))
```

Menu

Statistics > FMM (finite mixture models) > General estimation and regression

Syntax

Standard syntax

```
fmm # [ if ] [ in ] [ weight ] [ , fmmopts ] : component
```

Hybrid syntax

```
fmm [ if ] [ in ] [ weight ] [ , fmmopts ] : (component1) (component2) ...
```

where the standard syntax for *component* is

```
model depvar indepvars [ , options ]
```

the hybrid syntax for *component* is

```
model depvar indepvars [ , lcprob(varlist) options ]
```

model is an estimation command, and *options* are *model*-specific estimation options.

fmmopts	Description
Model	
`lcinvariant(`*pclassname*`)`	specify parameters that are equal across classes; default is `lcinvariant(none)`
`lcprob(`*varlist*`)`	specify independent variables for class probabilities
`lclabel(`*name*`)`	name of the categorical latent variable; default is `lclabel(Class)`
`lcbase(`#`)`	base latent class
`constraints(`*constraints*`)`	apply specified linear constraints
SE/Robust	
`vce(`*vcetype*`)`	*vcetype* may be `oim`, `opg`, `robust`, or `cluster` *clustvar*
Reporting	
`level(`#`)`	set confidence level; default is `level(95)`
`nocnsreport`	do not display constraints
`noheader`	do not display header above parameter table
`nodvheader`	do not display dependent variables information in the header
`notable`	do not display parameter table
display_options	control columns and column formats, row spacing, line width, display of omitted variables and base and empty cells, and factor-variable labeling
Maximization	
maximize_options	control the maximization process
`startvalues(`*svmethod*`)`	method for obtaining starting values; default is `startvalues(factor)`
`emopts(`*maxopts*`)`	control EM algorithm for improved starting values
`noestimate`	do not fit the model; show starting values instead
`collinear`	keep collinear variables
`coeflegend`	display legend instead of statistics

varlist may contain factor variables; see [U] **11.4.3 Factor variables**.

`by`, `collect`, `statsby`, and `svy` are allowed; see [U] **11.1.10 Prefix commands**.

`vce()` and weights are not allowed with the `svy` prefix; see [SVY] **svy**.

`fweights`, `iweights`, and `pweights` are allowed; see [U] **11.1.6 weight**.

`collinear` and `coeflegend` do not appear in the dialog box.

See [U] **20 Estimation and postestimation commands** for more capabilities of estimation commands.

pclassname	Description
`cons`	intercepts and cutpoints
`coef`	fixed coefficients
`errvar`	covariances of errors
`scale`	scaling parameters
`all`	all the above
`none`	none of the above; the default

Options

Model

`lcinvariant(`*pclassname*`)` specifies which parameters of the model are constrained to be equal across the latent classes; the default is `lcinvariant(none)`.

`lcprob(`*varlist*`)` specifies that the linear prediction for a given latent class probability include the variables in *varlist*. `lcinvariant()` has no effect on these parameters.

In the standard syntax, *varlist* is used in the linear prediction for each latent class probability.

In the hybrid syntax, specify `lcprob(`$varlist_i$`)` in $component_i$ to include $varlist_i$ in the linear prediction for the ith latent class probability. `lcprob()` is not allowed to be specified in *fmmopts* if it is being used in one or more *component* specifications.

In the hybrid syntax, if you specify `lcprob()` in the component that corresponds with the base latent class, the option is ignored.

`lclabel(`*name*`)` specifies a name for the categorical latent variable; the default is `lclabel(Class)`.

`lcbase(`#`)` specifies that # is to be treated as the base latent class.

In the standard syntax, the default is `lcbase(1)`.

In the hybrid syntax, the default base is the latent class corresponding to the first *component* that does not have `lcprob()` specified. If all components have `lcprob()`, the first *component* is the base and the `lcprob()` option specified for the first *component* is ignored.

`constraints()`; see [R] **Estimation options**.

SE/Robust

`vce(`*vcetype*`)` specifies the type of standard error reported, which includes types that are derived from asymptotic theory (`oim`, `opg`), that are robust to some kinds of misspecification (`robust`), and that allow for intragroup correlation (`cluster` *clustvar*); see [R] ***vce_option***.

Reporting

level(#); see [R] **Estimation options**.

nocnsreport suppresses the display of the constraints. Fixed-to-zero constraints that are automatically set by fmm are not shown in the report to keep the output manageable.

noheader suppresses the header above the parameter table, the display that reports the final log-likelihood value, number of observations, etc.

nodvheader suppresses the dependent variables information from the header above each parameter table.

notable suppresses the parameter tables.

display_options: noci, nopvalues, noomitted, vsquish, noemptycells, baselevels, allbaselevels, nofvlabel, fvwrap(#), fvwrapon(*style*), cformat(%*fmt*), pformat(%*fmt*), sformat(%*fmt*), and nolstretch; see [R] **Estimation options**.

Maximization

maximize_options: difficult, technique(*algorithm_spec*), iterate(#), [no]log, trace, gradient, showstep, hessian, tolerance(#), ltolerance(#), nrtolerance(#), nonrtolerance, and from(*init_specs*); see [R] **Maximize**. These options are seldom used.

startvalues() specifies how starting values are to be computed. Starting values specified in from() override the computed starting values.

startvalues(factor [, *maxopts*]) specifies that starting values are computed by assigning each observation to an initial latent class that is determined by running a factor analysis on all the observed variables in the specified model. This is the default.

startvalues(classid *varname* [, *maxopts*]) specifies that starting values are computed by assigning each observation to an initial latent class specified in *varname*. *varname* is required to have each class represented in the estimation sample.

startvalues(classpr *varlist* [, *maxopts*]) specifies that starting values are computed using the initial class probabilities specified in *varlist*. *varlist* is required to contain g variables for a model with g latent classes. The values in *varlist* are normalized to sum to 1 within each observation.

startvalues(randomid [, draws(#) seed(#) *maxopts*]) specifies that starting values are computed by randomly assigning observations to initial classes.

startvalues(randompr [, draws(#) seed(#) *maxopts*]) specifies that starting values are computed by randomly assigning initial class probabilities.

startvalues(jitter [$\#_c$ [$\#_v$], draws(#) seed(#) *maxopts*]) specifies that starting values are constructed by randomly perturbing the values from a Gaussian approximation to each outcome.

$\#_c$ is the magnitude for randomly perturbing coefficients, intercepts, cutpoints, and scale parameters; the default value is 1.

$\#_v$ is the magnitude for randomly perturbing variances for Gaussian outcomes; the default value is 1.

startvalues(zero) specifies that starting values are to be set to 0. This option is only useful if you use from() to specify starting values for some parameters and want the remaining starting values to be 0.

Most starting values options have suboptions that allow for tuning the starting values calculations:

maxopts is a subset of the standard *maximize_options*: difficult, technique(*algorithm_spec*), iterate(#), [no]log, trace, gradient, showstep, hessian, showtolerance, tolerance(#), ltolerance(#), and nrtolerance(#); see [R] **Maximize**.

draws(#) specifies the number of random draws. For startvalues(randomid), startvalues(randompr), and startvalues(jitter), fmm will generate # random draws and select the starting values from the draw with the best log-likelihood value from the EM iterations. The default is draws(1).

seed(#) sets the random-number seed.

emopts(*maxopts*) controls maximization of the log likelihood for the EM algorithm. *maxopts* is the same subset of *maximize_options* that are allowed in the startvalues() option, but some of the defaults are different for the EM algorithm. The default maximum number of iterations is iterate(20). The default coefficient vector tolerance is tolerance(1e-4). The default log-likelihood tolerance is ltolerance(1e-6).

noestimate specifies that the model is not to be fit. Instead, starting values are to be shown (as modified by the above options if modifications were made), and they are to be shown using the coeflegend style of output. An important use of this option is before you have modified starting values at all; you can type the following:

```
. fmm ..., ... noestimate : ...
. matrix b = e(b)
. ... (modify elements of b) ...
. fmm ..., ... from(b) : ...
```

The following options are available with fmm but are not shown in the dialog box:

collinear; see [R] **Estimation options**.

coeflegend displays the legend that reveals how to specify estimated coefficients in _b[] notation, which you are sometimes required to type when specifying postestimation commands.

Remarks and examples

For a general introduction to finite mixture models, see [FMM] **fmm intro**. For the list of estimation commands supported by the fmm prefix, see [FMM] **fmm estimation**.

Examples using fmm can be found at

[FMM] **Example 1a** — Mixture of linear regression models
[FMM] **Example 1b** — Covariates for class membership
[FMM] **Example 1c** — Testing coefficients across class models
[FMM] **Example 1d** — Component-specific covariates
[FMM] **Example 2** — Mixture of Poisson regression models
[FMM] **Example 3** — Zero-inflated models
[FMM] **Example 4** — Mixture cure models for survival data

Stored results

fmm stores the following in e():

Scalars

e(N)	number of observations
e(k)	number of parameters
e(k_eq)	number of equations in e(b)
e(k_dv)	number of dependent variables
e(k_cat#)	number of categories for the #th *depvar*, ordinal
e(k_out#)	number of categories for the #th *depvar*, mlogit
e(ll)	log likelihood
e(N_clust)	number of clusters
e(rank)	rank of e(V)
e(ic)	number of iterations
e(rc)	return code
e(converged)	1 if target model converged, 0 otherwise

Macros

e(cmd)	gsem
e(cmd2)	fmm
e(cmdline)	command as typed
e(prefix)	fmm
e(depvar)	names of dependent variables
e(eqnames)	names of equations
e(wtype)	weight type
e(wexp)	weight expression
e(title)	title in estimation output
e(clustvar)	name of cluster variable
e(model#)	model for the #th component
e(offset#)	offset for the #th *depvar*
e(vce)	*vcetype* specified in vce()
e(vcetype)	title used to label Std. err.
e(opt)	type of optimization
e(which)	max or min; whether optimizer is to perform maximization or minimization
e(method)	estimation method: ml
e(ml_method)	type of ml method
e(user)	name of likelihood-evaluator program
e(technique)	maximization technique
e(properties)	b V
e(estat_cmd)	program used to implement estat
e(predict)	program used to implement predict
e(covariates)	list of covariates
e(lclass)	name of latent class variable
e(marginsnotok)	predictions not allowed by margins
e(marginsdefault)	default predict() specification for margins
e(footnote)	program used to implement the footnote display
e(asbalanced)	factor variables fvset as asbalanced
e(asobserved)	factor variables fvset as asobserved

Matrices

e(b)	parameter vector
e(b_pclass)	parameter class
e(cat#)	categories for the #th *depvar*, ordinal
e(out#)	outcomes for the #th *depvar*, mlogit
e(Cns)	constraints matrix
e(ilog)	iteration log (up to 20 iterations)
e(gradient)	gradient vector
e(V)	covariance matrix of the estimators
e(V_modelbased)	model-based variance
e(lclass_k_levels)	number of levels for latent class variables

e(lclass_bases)	base levels for latent class variables
e(_N)	sample size for each component

Functions

e(sample)	marks estimation sample

In addition to the above, the following is stored in r():

Matrices

r(table)	matrix containing the coefficients with their standard errors, test statistics, p-values, and confidence intervals

Note that results stored in r() are updated when the command is replayed and will be replaced when any r-class command is run after the estimation command.

Methods and formulas

Methods and formulas are presented under the following headings:

The likelihood

fmm fits finite mixture models via maximum likelihood estimation. The likelihood for the specified model is derived under the assumption that, within a given latent class, each response variable is independent and identically distributed across the estimation sample. These assumptions are conditional on the latent classes and the observed exogenous variables.

The likelihood is computed by combining the conditional likelihoods from each latent class weighted by the associated latent-class probabilities. Let $\boldsymbol{\theta}$ be the vector of model parameters. For a given observation, let $\boldsymbol{y}$ be the vector of observed response variables, and $\boldsymbol{x}$ be the vector of independent variables. Let C be the categorical latent variable with g latent classes $1, \ldots, g$. The marginal likelihood for a given observation looks something like

$$\mathcal{L}_C(\boldsymbol{\theta}) = \sum_{i=1}^{g} \pi_i f_i(\boldsymbol{y}|\boldsymbol{x}, c_i = 1, \boldsymbol{\theta})$$

where π_i is the probability for the ith latent class, $f_i(\cdot)$ is the conditional probability density function for the observed response variables in the ith latent class, and $\boldsymbol{c}' = (c_1, \ldots, c_g)$ is the vector of latent class indicators. When $c_i = 1$, all other elements of $\boldsymbol{c}$ are zero. All auxiliary parameters are fit directly without any further parameterization, so we simply acknowledge that the auxiliary parameters are among the elements of $\boldsymbol{\theta}$.

The $\boldsymbol{y}$ variables are assumed to be independent, conditional on $\boldsymbol{x}$ and C, so $f_i(\cdot)$ is the product of the individual conditional densities. One exception to this is when $\boldsymbol{y}$ contains the outcome and endogenous covariates for ivregress, in which case the Gaussian responses are actually modeled using a multivariate normal density to allow for correlated errors. This one exception does not meaningfully change the following discussion, so we make no effort to represent this distinction in the formulas.

For the ith latent class with n response variables, the conditional joint density function for a given observation is

$$f_i(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{\theta}) = \prod_{j=1}^{n} f_{ij}(y_{ij}|\boldsymbol{x}, \boldsymbol{\theta})$$

All estimation commands supported by fmm model the dependence of y_{ij} on $\boldsymbol{x}$ through the linear prediction

$$z_{ij} = \boldsymbol{x}'\boldsymbol{\beta}_{ij}$$

where $\boldsymbol{\beta}_{ij}$ is the vector of the coefficients for y_{ij}. For notational convenience, we will overload the definitions of $f_i(\cdot)$ and $f_{ij}(\cdot)$ so that they are functions of the responses and model parameters through the linear predictions $\boldsymbol{z}_i' = (z_{i1}, \ldots, z_{in})$. Thus $f_i(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{\theta})$ is equivalently specified as $f_i(\boldsymbol{y}, \boldsymbol{z}_i, \boldsymbol{\theta})$, and $f_{ij}(y_{ij}|\boldsymbol{x}, \boldsymbol{\theta})$ is equivalently specified as $f_{ij}(y_{ij}, z_{ij}, \boldsymbol{\theta})$. In this new notation, the likelihood for a given observation is

$$\mathcal{L}(\boldsymbol{\theta}) = \sum_{i=1}^{g} \pi_i \prod_{j=1}^{n} f_{ij}(y_{ij}, z_{ij}, \boldsymbol{\theta}) \tag{1}$$

fmm uses the multinomial logistic distribution to model the probabilities for the latent classes. For the ith latent class, the probability is given by

$$\pi_i = \Pr(c_i = 1|\boldsymbol{x}) = \frac{\exp(z_i)}{\sum_{j=1}^{g} \exp(z_j)}$$

where the linear prediction for the ith latent class is

$$z_i = \boldsymbol{x}'\boldsymbol{\gamma}_i$$

and $\boldsymbol{\gamma}_i$ is the associated vector of coefficients. If the first latent class is the base level, $\boldsymbol{\gamma}_1$ is a vector of zeros so that $z_1 = 0$ and $\exp(z_1) = 1$.

The vector $\boldsymbol{\theta}$ is therefore the set of unique model parameters taken from the following:

$\boldsymbol{\gamma}_i$ is the vector of coefficients for the ith latent class.

$\boldsymbol{\beta}_{ij}$ is the vector of coefficients for y_{ij}.

Auxiliary parameters are parameters that result from some of the distribution families.

Each latent class will have its own set of these parameters.

The EM algorithm

fmm uses the EM algorithm to refine starting values before maximizing the likelihood in (1).

The EM algorithm uses the complete-data likelihood, a likelihood where it is as if we have observed values for the latent class indicator variables $\boldsymbol{c}$. In the complete-data case, the likelihood for a given observation is

$$L(\boldsymbol{\theta}) = \prod_{i=1}^{g} \{\pi_i f_i(\boldsymbol{y}, \boldsymbol{z}_i, \boldsymbol{\theta})\}^{c_i}$$

so the complete-data log likelihood is

$$\log L(\boldsymbol{\theta}) = \sum_{i=1}^{g} c_i \{\log \pi_i + \log f_i(\boldsymbol{y}, \boldsymbol{z}_i, \boldsymbol{\theta})\}$$

We intend to maximize the expected complete-data log likelihood given the observed variables $\boldsymbol{y}$ and $\boldsymbol{x}$. This is an iterative process in which we use the kth guess of the model parameters, denoted $\boldsymbol{\theta}_{(k)}$, then compute the next guess, $\boldsymbol{\theta}_{(k+1)}$.

In the expectation (E) step, we derive the functional form of the expected complete-data log likelihood. The complete-data log likelihood is a linear function of the latent class indicator variables, so

$$\mathrm{E}(c_i|\boldsymbol{y},\boldsymbol{x},\boldsymbol{\theta}_{(k)}) = \frac{\pi_i f_i(\boldsymbol{y},\boldsymbol{z}_i,\boldsymbol{\theta}_{(k)})}{\sum_{j=1}^{g}\pi_j f_j(\boldsymbol{y},\boldsymbol{z}_j,\boldsymbol{\theta}_{(k)})}$$

We denote this posterior probability by p_i, so the expected complete-data log likelihood for a given observation is given by

$$Q(\boldsymbol{\theta}|\boldsymbol{\theta}_{(k)}) = \sum_{i=1}^{g} p_i \left\{ \log \pi_i + \log f_i(\boldsymbol{y},\boldsymbol{z}_i,\boldsymbol{\theta}) \right\}$$

Note that $Q(\boldsymbol{\theta}|\boldsymbol{\theta}_{(k)})$ is a function of $\boldsymbol{\theta}_{(k)}$ solely through the posterior probabilities p_i.

Now that we have the conditional complete-data log likelihood, the maximization (M) step is to maximize $Q(\boldsymbol{\theta}|\boldsymbol{\theta}_{(k)})$ with respect to $\boldsymbol{\theta}$ to find $\boldsymbol{\theta}_{(k+1)}$.

Survey data

`fmm` supports estimation with survey data. However, only the linearized variance estimator is supported. For details on VCEs with survey data, see [SVY] **Variance estimation**.

Predictions

The predicted mean for a given response within a latent class is computed in the standard way. For example, the predicted mean for `regress` is the linear prediction, the predicted mean for `glm` is computed by applying the link function to the linear prediction, and for `ologit`, the predicted mean for a given response level is the predicted probability for that level. For survival outcomes, the formulas for predicted means (expected values) are provided in the *Survival distributions* section in [SEM] **Methods and formulas for gsem**.

Let $\hat{z}_i$ be the linear prediction for the ith latent class. The predicted probability for the ith latent class is then given by

$$\hat{\pi}_i = \frac{\exp(\hat{z}_i)}{\sum_{j=1}^{g}\exp(\hat{z}_j)}$$

The predicted posterior probability for the ith latent class is given by

$$\tilde{\pi}_i = \frac{\hat{\pi}_i f_i(\boldsymbol{y},\hat{\boldsymbol{z}}_i,\hat{\boldsymbol{\theta}})}{\sum_{j=1}^{g}\hat{\pi}_j f_j(\boldsymbol{y},\hat{\boldsymbol{z}}_j,\hat{\boldsymbol{\theta}})}$$

Let $\hat{\mu}_i$ be the predicted mean of response y in the ith latent class. The predicted overall mean of y, using the fitted latent class probabilities, is given by

$$\hat{\mu} = \sum_{i=1}^{g}\hat{\pi}_i\hat{\mu}_i$$

The predicted overall mean of y, using the posterior latent class probabilities, is given by

$$\tilde{\mu} = \sum_{i=1}^{g}\tilde{\pi}_i\hat{\mu}_i$$

Also see

[FMM] **fmm intro** — Introduction to finite mixture models

[FMM] **fmm estimation** — Fitting finite mixture models

[FMM] **fmm postestimation** — Postestimation tools for fmm

[FMM] **Glossary**

[SVY] **svy estimation** — Estimation commands for survey data