

# Modeling unobserved heterogeneity in Stata

Rafal Raciborski

StataCorp LLC

November 27, 2017

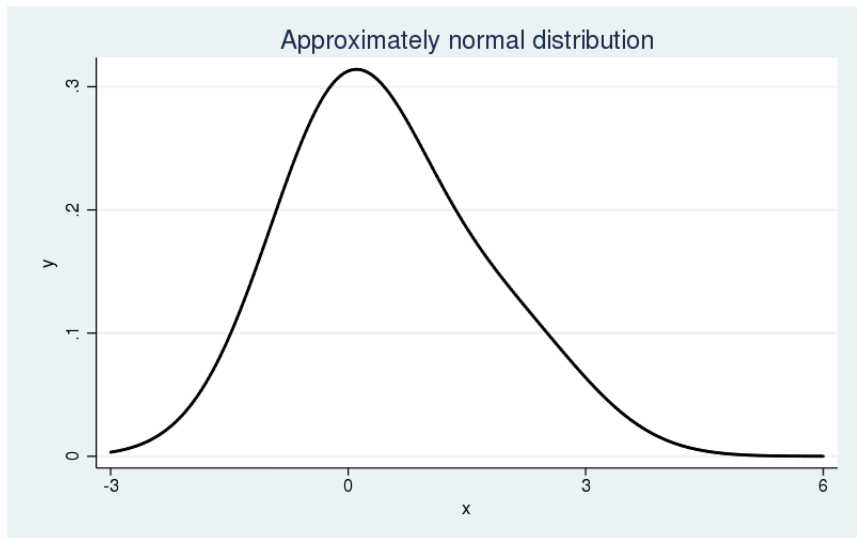
## Plan of the talk

Concepts and terminology

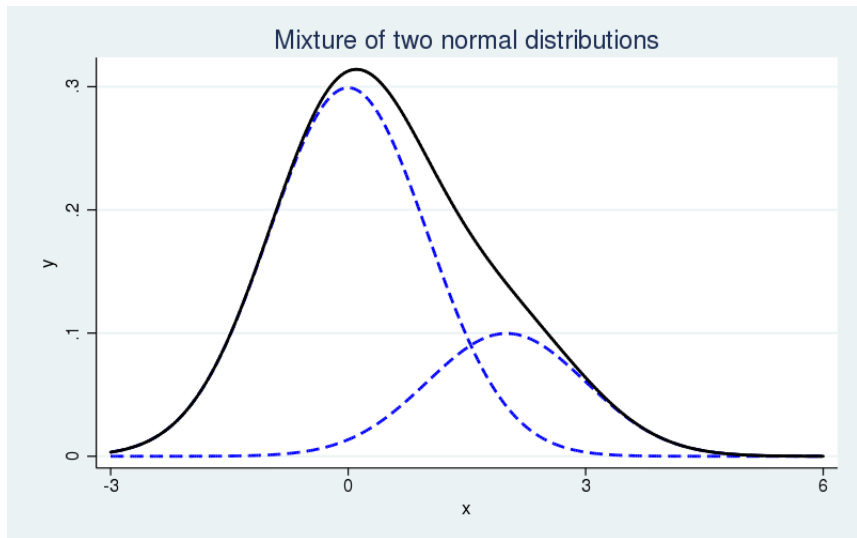
Finite mixture models with **fmm**

Latent class models with **gsem** . . . **lclass()**

Observed distribution for a whole population:



Unobserved distributions of the two underlying subpopulations:



Unobserved heterogeneity refers to differences among individuals or observations that cannot be measured by regressors.

Latent class models

Latent – unobserved, hidden

Class – subpopulation, group, type, component, density, distribution

Finite mixture models

Finite – number of classes determined a priori

Mixture – of distributions, densities, regression models

Mixture of distributions:

The observed  $\mathbf{y}$  are assumed to come from  $g$  distinct distributions  $f_1, f_2, \dots, f_g$  in proportions or with probabilities  $\pi_1, \pi_2, \dots, \pi_g$ .

We can write a simple mixture model as

$$f(\mathbf{y}) = \sum_{i=1}^g \pi_i f_i(\mathbf{y} | \mathbf{x}'\beta_i)$$

where  $\pi_i$  is the probability for the  $i$ th class, and  $f_i(\cdot)$  is the conditional probability density function (pdf) for the observed response in the  $i$ th class model.



(continued)

$$f(\mathbf{y}) = \sum_{i=1}^g \pi_i f_i(\mathbf{y} | \mathbf{x}'\beta_i)$$

We use the multinomial logistic distribution to model the probabilities for the latent classes.

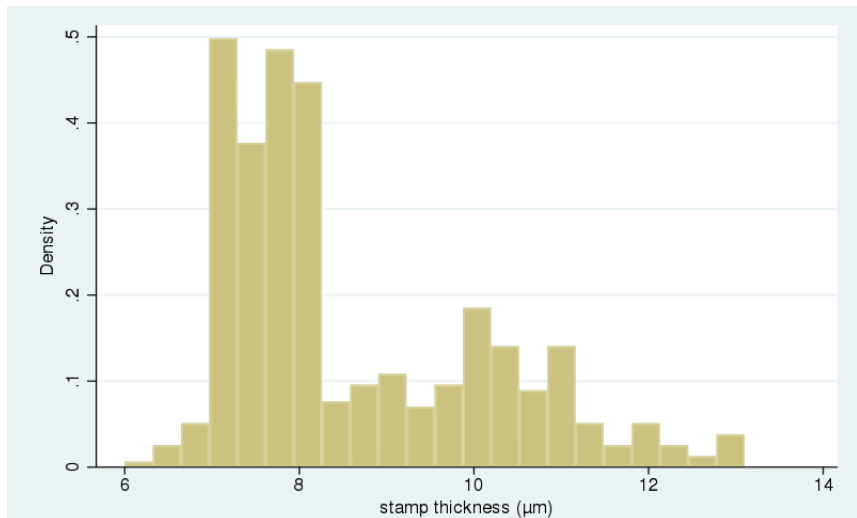
$$\pi_i = \frac{\exp(\gamma_i)}{\sum_{j=1}^g \exp(\gamma_j)}$$

where  $\gamma_i$  is the linear prediction for the  $i$ th latent class.

By convention, the first latent class is the base category,  $\gamma_1 = 0$ .

## Example: Postal stamp thickness

```
. webuse stamp  
. gen thick = thickness*100  
. label var thick "stamp thickness ( $\mu\text{m}$ )"  
. histogram thick
```



We want to model the empirical distribution as a mixture of two normal distributions:

$$f(\mathbf{y}) = \pi_1 \times N(\mu_1, \sigma_1^2) + \pi_2 \times N(\mu_2, \sigma_2^2)$$

This is as simple as typing:

```
. fmm 2 : regress thick
```

where

**fmm 2** means we have two components

and **regress** is a keyword for “normal distribution”



Class : 1  
Response : thick  
Model : regress

```
-----+-----  
            |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]  
-----+-----  
thick      |  
    _cons |    7.609076   .0297275   255.96   0.000    7.550811    7.667341  
-----+-----  
var(e.thick) |    .206297   .022201                .1670665    .2547395  
-----+-----
```

Class : 2  
Response : thick  
Model : regress

```
-----+-----  
            |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]  
-----+-----  
thick      |  
    _cons |   10.16013   .1427942   71.15   0.000    9.880254   10.44  
-----+-----  
var(e.thick) |    1.441319   .2583438                1.014354    2.048003  
-----+-----
```

Recall we use the multinomial logistic distribution to model the probabilities for the latent classes:

$$\pi_i = \frac{\exp(\gamma_i)}{\sum_{j=1}^g \exp(\gamma_j)}$$

In simple cases, we can calculate latent class probabilities by hand:

```
. di          1 / ( 1 + exp(-.4498027) )  
. di exp(-.4498027) / ( 1 + exp(-.4498027) )  
  
.61059232  
.38940768
```

This is a little bit easier:

```
. di          1 / ( 1 + exp(_b[2.Class:_cons]) )  
. di exp(_b[2.Class:_cons]) / ( 1 + exp(_b[2.Class:_cons]) )  
  
.61059232  
.38940768
```

You can also use **predict** and **summarize**:

```
. predict pr*, classposteriorpr
```

```
. des pr1 pr2
```

variable name	storage type	display format	value label	variable label
pr1	float	%9.0g		Predicted posterior probability (1.Class)
pr2	float	%9.0g		Predicted posterior probability (2.Class)

```
. su pr1 pr2
```

Variable	Obs	Mean	Std. Dev.	Min	Max
pr1	485	.6105923	.4519458	1.53e-30	.9829751
pr2	485	.3894077	.4519458	.0170249	1



## estat lcprob is your friend:

```
. estat lcprob
```

```
Latent class marginal probabilities          Number of obs      =          485
```

---

		Delta-method		
	Margin	Std. Err.	[95% Conf. Interval]	
Class				
1	.6105923	.0295055	.5514633	.6666385
2	.3894077	.0295055	.3333615	.4485367

---

Note that when you have a mixture of distributions, the posterior probability of being in a given class is the same for all observations with the same value.

```
. su pr1 pr2 if thick==8
```

Variable	Obs	Mean	Std. Dev.	Min	Max
pr1	37	.93524	0	.93524	.93524
pr2	37	.06476	0	.06476	.06476

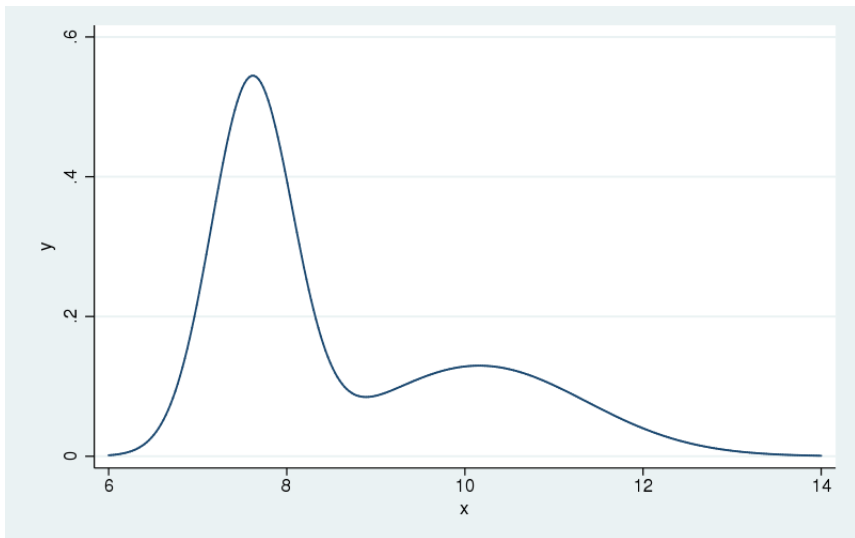
This makes it easy to plot the estimated mixture density.

This is our estimated mixture density:

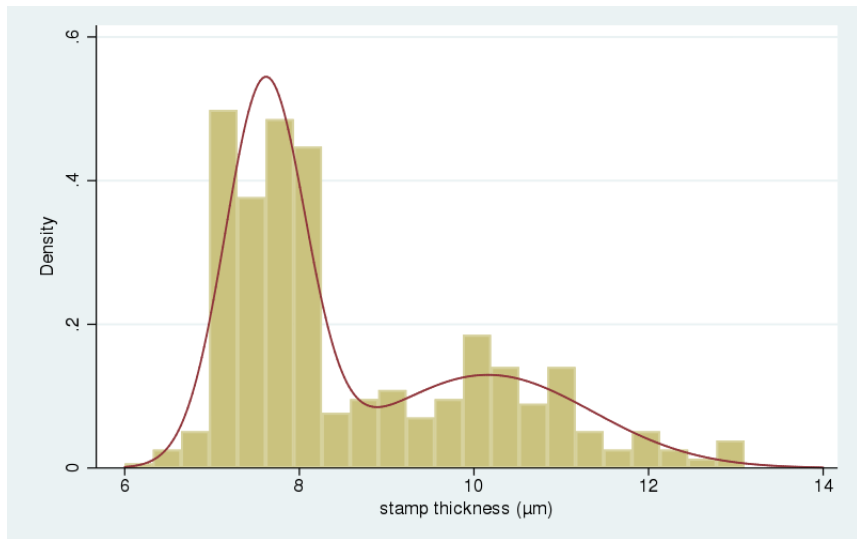
$$\hat{f}(\mathbf{y}) = .61 \times N(7.61, .21) + .39 \times N(10.16, 1.44)$$

```
. twoway ///
```

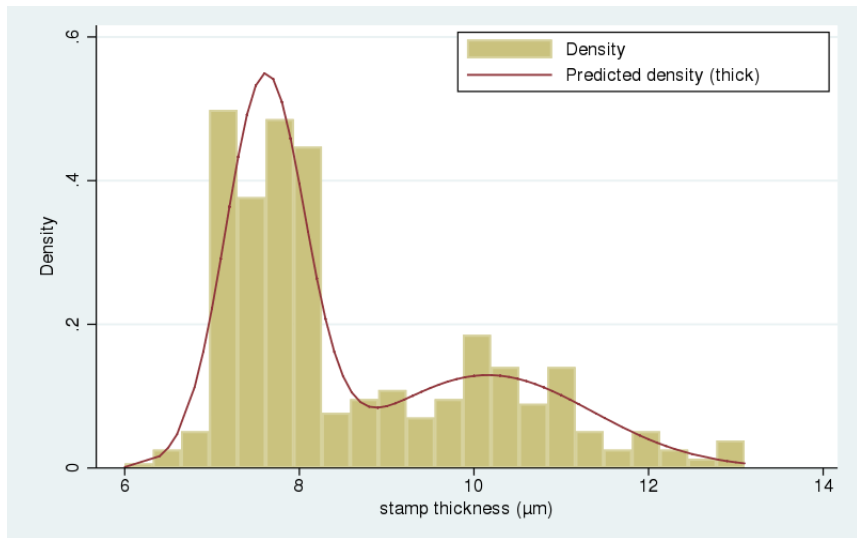
```
function .61*normalden(x,7.61,sqrt(.21)) + .39*normalden(x,10.16,sqrt(1.44)), range(6 14)
```



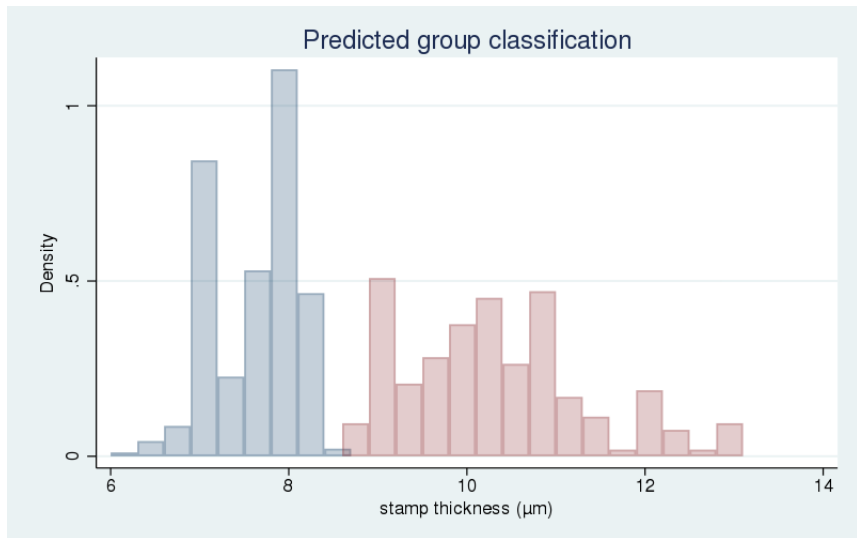
```
. histogram thick, addplot( ///  
    function .61*normalden(x,7.61,sqrt(.21)) + .39*normalden(x,10.16,sqrt(1.44)) range(6 14) ///  
    ) legend(off)
```



```
. predict den, density marginal  
. histogram thick, addplot(line den thick) legend(ring(0) pos(2))
```



```
. gen group = pr1 > .5  
. twoway histogram thick if group ... ///  
    histogram thick if !group ...
```



When we add covariates, we fit a mixture of “models”.

Here, we fit a mixture of two linear regression models.

```
. use chol
```

```
(Fictional cholesterol data)
```

```
. describe
```

variable name	storage type	display format	value label	variable label
chol	float	%9.0g		Standardized cholesterol level
wine	float	%9.0g		Mean-centered monthly wine consumption
pchol	float	%9.0g		=1 if either parent has high cholesterol level

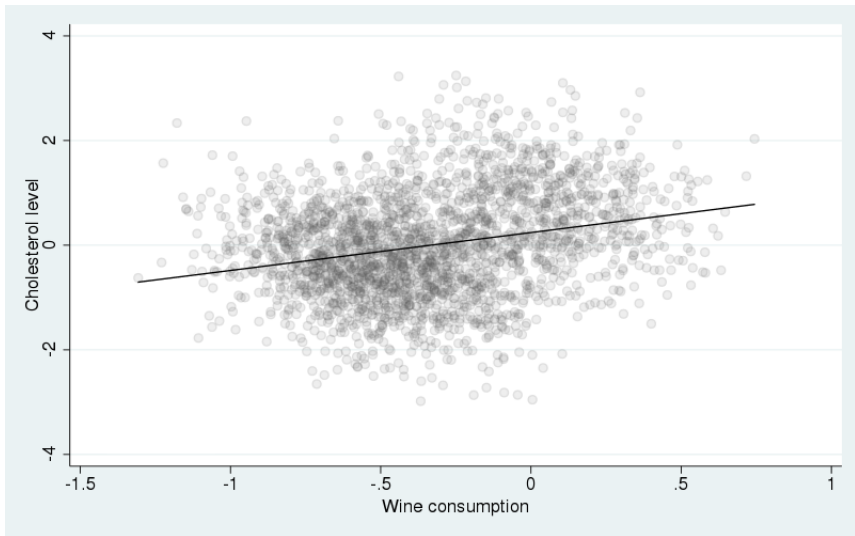


```
. regress chol wine
```

Source	SS	df	MS	Number of obs	=	2,500
-----+-----				F(1, 2498)	=	171.27
Model	160.343489	1	160.343489	Prob > F	=	0.0000
Residual	2338.65652	2,498	.936211577	R-squared	=	0.0642
-----+-----				Adj R-squared	=	0.0638
Total	2499.00001	2,499	1	Root MSE	=	.96758

chol	Coef.	Std. Err.	t	P> t	[95% Conf. Interval]	
-----+-----						
wine	.7243775	.0553511	13.09	0.000	.6158387	.8329162
_cons	.2408989	.0267081	9.02	0.000	.1885266	.2932712
-----						

```
. twoway (scatter chol wine ...) (lfit chol wine ...) ...
```



```
. fmm 2, lcpob(pchol): regress chol wine
```

```
Finite mixture model                Number of obs    =      2,500  
Log likelihood = -3062.7143
```

	Coef.	Std. Err.	z	P> z	[95% Conf. Interval]	
-----+-----						
1.Class	(base outcome)					
-----+-----						
2.Class						
pchol	7.473592	.8977705	8.32	0.000	5.713994	9.23319
_cons	-3.228661	.3939579	-8.20	0.000	-4.000804	-2.456518
-----						

Class : 1  
 Response : chol  
 Model : regress

	Coef.	Std. Err.	z	P> z	[95% Conf. Interval]	
-----+-----						
chol						
wine	-.6850974	.0783981	-8.74	0.000	-.8387549	-.5314399
_cons	-.7401758	.0443478	-16.69	0.000	-.8270959	-.6532557
-----+-----						
var(e.chol)	.6152073	.0219867			.5735887	.6598457
-----						

Class : 2  
 Response : chol  
 Model : regress

	Coef.	Std. Err.	z	P> z	[95% Conf. Interval]	
-----+-----						
chol						
wine	-.4798618	.1319125	-3.64	0.000	-.7384056	-.221318
_cons	.8343004	.0323813	25.76	0.000	.7708342	.8977667
-----+-----						
var(e.chol)	.6720669	.0383181			.601009	.7515261
-----						

```
. predict c*, classposteriorpr
. su c?
```

Variable	Obs	Mean	Std. Dev.	Min	Max
c1	2,500	.6743291	.4538173	6.22e-09	1
c2	2,500	.3256709	.4538173	4.24e-11	1

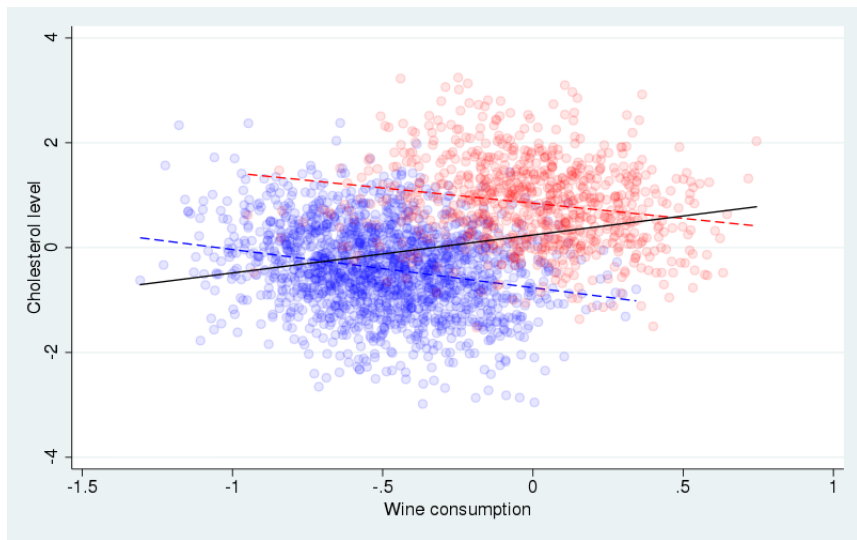
```
. estat lcprob
```

Latent class marginal probabilities                      Number of obs        =        2,500

Class	Delta-method			
	Margin	Std. Err.	[95% Conf. Interval]	
1	.6743291	.0055936	.6632719	.6851956
2	.3256709	.0055936	.3148044	.3367281

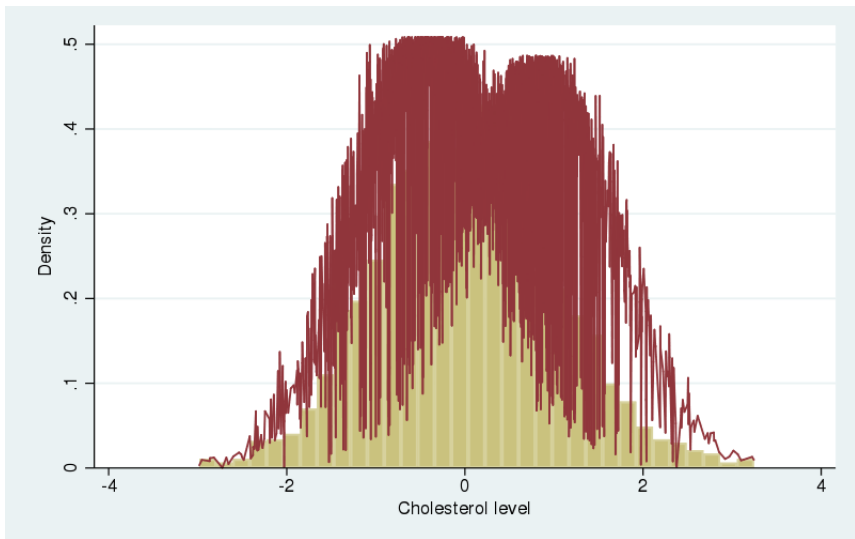


```
. gen grp = c1 > .5  
. twoway (scatter chol wine if grp ...) (scatter chol wine if !grp ...) ///  
      ( lfit chol wine if grp ...) ( lfit chol wine if !grp ...) ///  
      ( lfit chol wine ...) ...
```



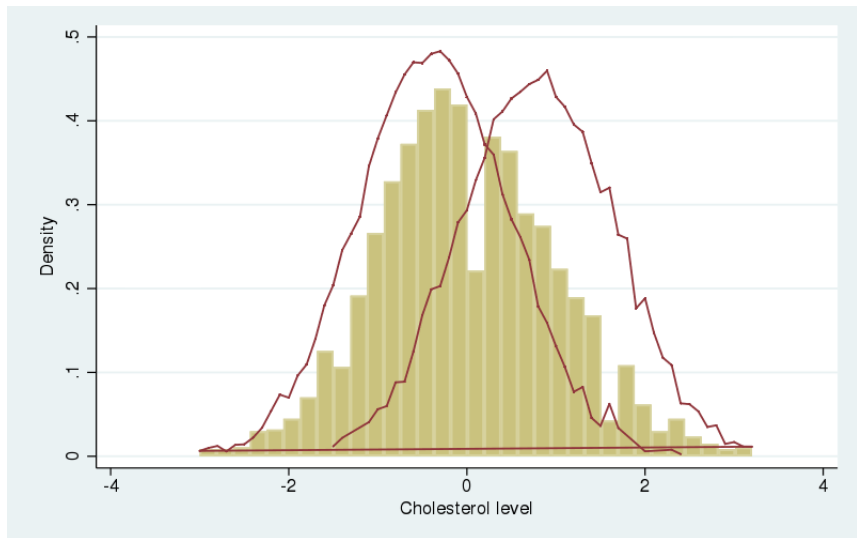
## Some tips ...

```
. predict den, density marginal  
. sort chol  
. histogram chol, addplot(line den chol) legend(off)
```

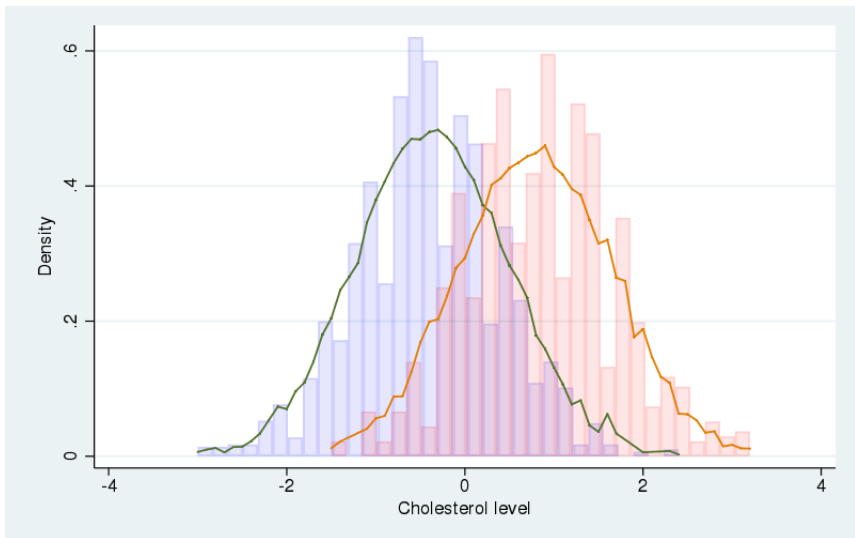




```
. gen chol2 = round(chol,.1)
. sort grp chol2
. by grp chol2 : egen den2 = mean(den)
. twoway (histogram chol2) (line den2 chol2) ...
```



```
. twoway ///  
>     (histogram chol2 if grp, color(blue%10)) (line den2 chol2 if grp) ///  
>     (histogram chol2 if !grp, color(red%10)) (line den2 chol2 if !grp) ...
```



**regress** is one of the many **fmm** keywords

Linear outcomes: **regress**, **truncreg**, **intreg**, **tobit**, and **ivregress**

Binary outcomes: **logit**, **probit**, and **cloglog**

Ordinal outcomes: **ologit** and **oprobit**

Nominal outcomes: **mlogit**

Count outcomes: **poisson**, **nbreg**, and **tpoisson**

Generalized linear models: **glm** with family **beta**, **exponential**, **gamma**, **lognormal**, and more

Fractional outcomes: **betareg**

Survival outcomes: **streg**

**fmm** has two syntaxes.

You have seen the simple syntax

```
. fmm 3 : regress y x1 x2 x3
```

You can also use the hybrid syntax

```
. fmm : (regress y x1 x2 x3) (regress y x1 x2 x3) (regress y x1 x2 x3)
```

In the simple syntax, each component gets the same regressors:

```
. fmm 3 : regress y x1 x2 x3  
. fmm   : (regress y x1 x2 x3) (regress y x1 x2 x3) (regress y x1 x2 x3)
```

In the hybrid syntax, each component can have different regressors

```
. fmm : (regress y x1 x2) (regress y x2 x3) (regress y x3, noconstant)
```

In the hybrid syntax, you can also fit mixtures of different models or distributions:

```
. fmm : (regress y) (glm y, family(lognormal)) (tobit y, ll(0))
```

```
. fmm : (regress y x1) (glm y, family(lognormal)) (tobit y x1 x2, ll(0))
```

You can also model the latent class probabilities using the **lcprob()** option.

## Simple syntax

```
. fmm 3, lcprob(w1 w2) : regress y x1 x2 x3
```

## Hybrid syntax

```
. fmm : (regress y x1) (regress y x2 x3, lcprob(w1 w2)) (regress y x1 x3, lcprob(w2))
```

There is one special **fmm** keyword **pointmass** that allows one or more components to be a degenerate distribution taking on a single integer value with probability one.

This distribution cannot be used by itself and is always combined with other **fmm** keywords, most often to model zero-inflated outcomes.

This means you can use **fmm** in place of **zip** and **zinb**, and as an alternative to **zioprobit**.



## Example: zero-inflated Poisson model

The zero-inflated Poisson model is a model in which the distribution of the outcome is a two-component mixture. One component is a distribution that is all zero. The other component is a Poisson distribution.

```
. webuse fish, clear
```

```
. describe
```

```
-----
```

variable name	storage type	display format	value label	variable label
livebait	byte	%9.0g		1 if visitor uses live bait
camper	byte	%9.0g		1 if visitor is camping
persons	byte	%9.0g		number of persons accompanying the visitor
child	byte	%9.0g		number of children accompanying the visitor
count	int	%9.0g		number of fish caught

```
-----
```

```
. su count
```

```
-----
```

Variable	Obs	Mean	Std. Dev.	Min	Max
count	250	3.296	11.63503	0	149

```
-----
```

```
. zip count persons livebait, inflate(child camper)

. fmm : (poisson count persons livebait) (pointmass count, lprob(child camper))
```

Finite mixture model  
Log likelihood = -850.70142

Number of obs = 250

	Coef.	Std. Err.	z	P> z	[95% Conf. Interval]	
-----						
1.Class						
child	1.602571	.2797719	5.73	0.000	1.054228	2.150913
camper	-1.015698	.365259	-2.78	0.005	-1.731593	-.2998039
_cons	-.4922872	.3114562	-1.58	0.114	-1.10273	.1181558

2.Class       | (base outcome)

Class        : 2  
Response     : count  
Model        : poisson

	Coef.	Std. Err.	z	P> z	[95% Conf. Interval]	
-----						
count						
persons	.8068853	.0453288	17.80	0.000	.7180424	.8957281
livebait	1.757289	.2446082	7.18	0.000	1.277866	2.236713
_cons	-2.178472	.2860289	-7.62	0.000	-2.739078	-1.617865

```
. estat lcprob
```

```
Latent class marginal probabilities          Number of obs      =          250
```

	Margin	Delta-method Std. Err.	[95% Conf. Interval]	
Class				
1	.4786335	.0341083	.4125554	.5454678
2	.5213665	.0341083	.4545322	.5874446

```
. estat lcmean
```

```
Latent class marginal means          Number of obs      =          250
```

```
Expression : Predicted mean (number of fish caught in class 2.Class),  
            predict(outcome(count) class(2))
```

	Margin	Delta-method Std. Err.	z	P> z	[95% Conf. Interval]	
2						
count	6.490014	.2361623	27.48	0.000	6.027144	6.952884

With **pointmass()** you can run any imaginable inflated model (not that they all make sense though)

```
. fmm : (poisson y x1) (poisson y x2) (pointmass y)
```

```
. fmm : (poisson y x1) (poisson y x2) (pointmass y) (pointmass y, value(5))
```

```
. fmm : (ologit y) (pointmass y, value(1))
```

```
. fmm : (ologit y) (pointmass y, value(2)) (pointmass y, value(4))
```

```
. fmm : (mlogit y x1 x2 x3) (pointmass y, value(3))
```

With **fmm** you cannot fit mixture models for multiple responses. In other words, each class can have only one dependent variable.

```
. use chol, clear
. fmm : (regress chol) (regress wine)
Class      : 1
Response   : chol
Model      : regress
```

	Coef.	Std. Err.	z	P> z	[95% Conf. Interval]	
chol						
_cons	-.0450922	848.3992	-0.00	1.000	-1662.877	1662.787
var(e.chol)	.0584758	206.5966			.	.

```
Class      : 2
Response   : wine
Model      : regress
```

	Coef.	Std. Err.	z	P> z	[95% Conf. Interval]	
wine						
_cons	-.3325602	.0069923	-47.56	0.000	-.3462649	-.3188556
var(e.wine)	.1222298	.0034571			.1156383	.129197

If you want each class to have two outcomes, **chol** and **wine**, you need to go through **gsem**.

FMM in **gsem**:

A model with categorical latent variables and categorical observed variables is called a latent class model.

A model with categorical latent variables and continuous observed variables is called a latent profile model.

A finite mixture model can be either.

Start with a mixture of two normal distributions.

```
fmm 2: regress chol
```

Different ways of doing the same in **gsem**:

```
gsem (chol <-) , lclass(Class 2) lcinvariant(none)
```

```
gsem (chol <-) (chol <-) , lclass(Class 2) lcinvariant(none)
```

```
gsem (1: chol <-) (2: chol <-) , lclass(Class 2) lcinvariant(none)
```



The default model is linear regression so these are equivalent:

```
fmm 2: regress chol
```

```
gsem (chol <-) , lclass(Class 2) lcinvariant(none)
```

```
gsem (chol <-, regress) , lclass(Class 2) lcinvariant(none)
```

```
gsem (chol <-, family(gaussian)) , lclass(Class 2) lcinvariant(none)
```

```
gsem (chol <-, family(gaussian) link(identity)) , lclass(Class 2) lcinvariant(none)
```

Adding covariates is easy.

Below, both class models receive **the same** covariates:

```
fmm 2: regress chol wine
```

```
gsem (chol <- wine) , lclass(Class 2) lcinvariant(none)
```

```
gsem (chol <- wine) (chol <- ) , lclass(Class 2) lcinvariant(none)
```

You have to be more explicit if you want the model for class 2 to have only the constant term:

```
fmm : (regress chol wine) (regress chol)
```

```
gsem (1: chol <- wine) (2: chol <- ) , lclass(Class 2) lcinvariant(none)
```

Now you should be able to figure out how to fit a mixture model for multiple responses.

Pseudo-**fmm** syntax:

```
fmm 2: regress chol wine =
```

**gsem** syntax (real):

```
gsem (chol wine <-), lclass(Class 2) lcinvariant(none)
```

```
gsem (chol <-) (wine <-), lclass(Class 2) lcinvariant(none)
```

If you want class 1 to have two dependent variables and class 2 to have one dependent variable, you have to be explicit about it:

```
gsem (1: chol wine <-) (2: chol <- pchol), lclass(Class 2) lcinvariant(none)
```

Generalized structural equation model                      Number of obs        =        2,500  
Log likelihood = -3217.3559

	Coef.	Std. Err.	z	P> z	[95% Conf. Interval]	
1.Class	(base outcome)					
2.Class						
_cons	2.817545	.2573915	10.95	0.000	2.313067	3.322023

Class : 1  
 Response : chol  
 Family : Gaussian  
 Link : identity

Response : wine  
 Family : Gaussian  
 Link : identity

	Coef.	Std. Err.	z	P> z	[95% Conf. Interval]	
-----+-----						
chol						
_cons	-.7664867	.131116	-5.85	0.000	-1.023469	-.5095042
-----+-----						
wine						
_cons	-.4772524	.0243019	-19.64	0.000	-.5248833	-.4296215
-----+-----						
var(e.chol)	.2887896	.1144431			.1328198	.6279142
var(e.wine)	.0100204	.0032932			.0052619	.0190822
-----+-----						

Class : 2

Response : chol

Family : Gaussian

Link : identity

---

	Coef.	Std. Err.	z	P> z	[95% Conf. Interval]	
chol						
pchol	.4683264	.0184701	25.36	0.000	.4321258	.5045271
_cons	.0322942	.0206101	1.57	0.117	-.0081007	.0726892
var(e.chol)	.7804484	.0254232			.7321771	.8319021

---

For zero-inflated models in **gsem**, use **family(pointmass)**:

```
webuse fish, clear
```

```
fmm : (pointmass count) (poisson count persons livebait)
```

```
gsem ///
```

```
    (1: count <-           , family(pointmass)) ///
```

```
    (2: count <- persons livebait, family(poisson)) ///
```

```
    , ///
```

```
    lclass(Class 2) lcinvariant(none)
```



To model class probabilities, add a class equation:

```
fmm : (pointmass count, lcprob(child camper)) (poisson count persons livebait)
gsem ///
      (1: count <-,                family(pointmass)) ///
      (2: count <- persons livebait, family(poisson)) ///
      (Class <- child camper) ///
      , ///
      lclass(Class 2) lcinvariant(none)
```

With two classes, you can specify which class receives the predictors:

```
(1.Class <- child camper)
```

With more than two classes, you can specify different predictors for different classes:

```
(2.Class <- x1 x2) (3.Class <- x2 x3) ...
```

Last but not least, **gsem** allows you to specify more than one categorical latent variable:

```
gsem (x1 x2 x3 <- _cons), logit lclass(C 2) lclass(D 3)
```

In expanded notation, you get terms such as

```
(2.C#3.D: x1 <- _cons)
```

Questions?