Finite mixture models using Stata

Webinar by Rafal Raciborski

StataCorp LLC

June 19, 2018

Plan of the talk

Concepts and terminology

Finite mixture models with **fmm**

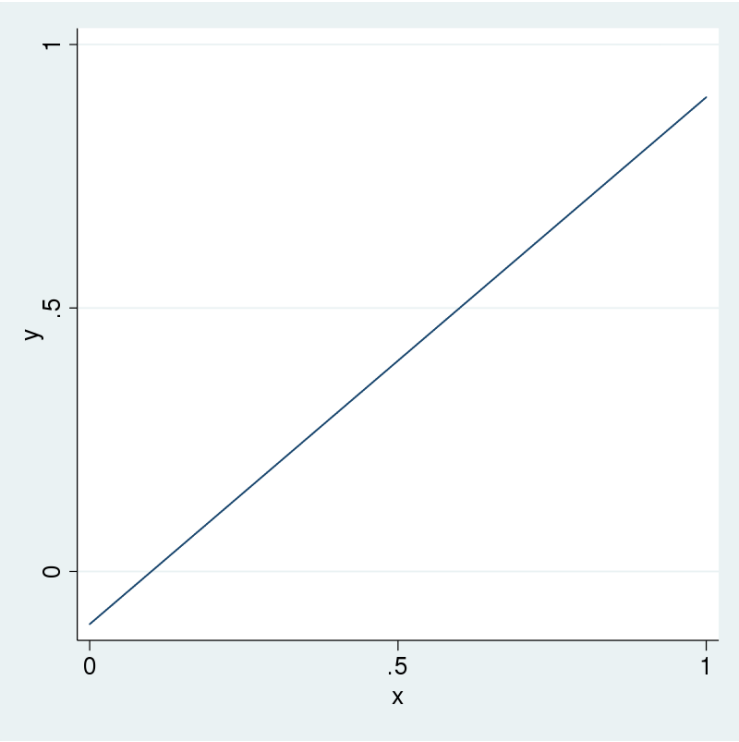
Latent class models with **gsem, lclass()**

FMM stands for finite mixture models.

Why do we use them? Let's think about fitting a regression model. If you have data created by a single process, we can fit a single regression model. In Stata, we might type

```
. regress y x
```

and we might get a regression model where the predictions look like this:

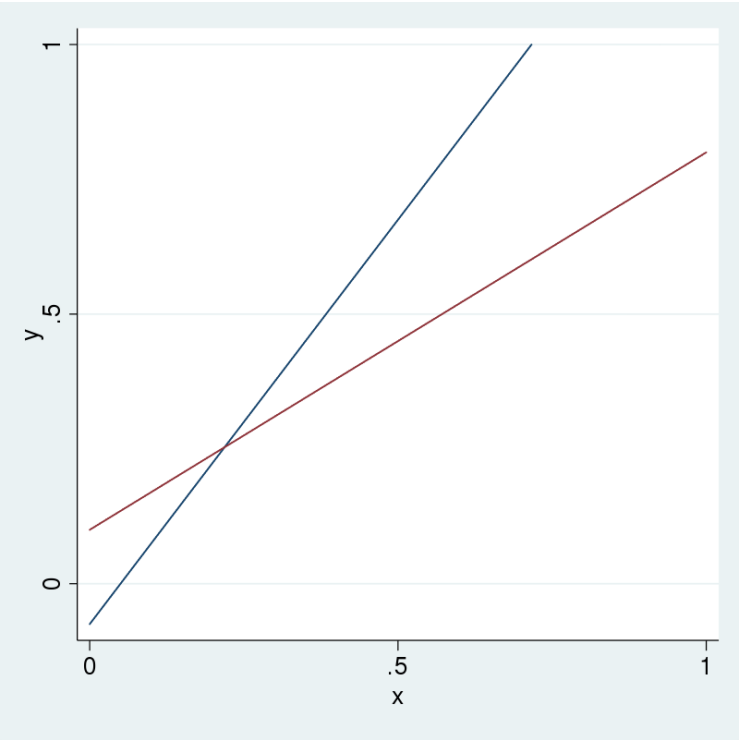


What if you have two or more subpopulations and regression parameters differ across them?

If you have an identifier that tells you which subpopulation each observation belongs to, you might include interaction terms in your model. In Stata, we might type

```
. regress y c.x##i.group
```

We could fit this model, and get separate regression lines for each group. For instance, we might have:



What if we cannot identify the underlying groups?

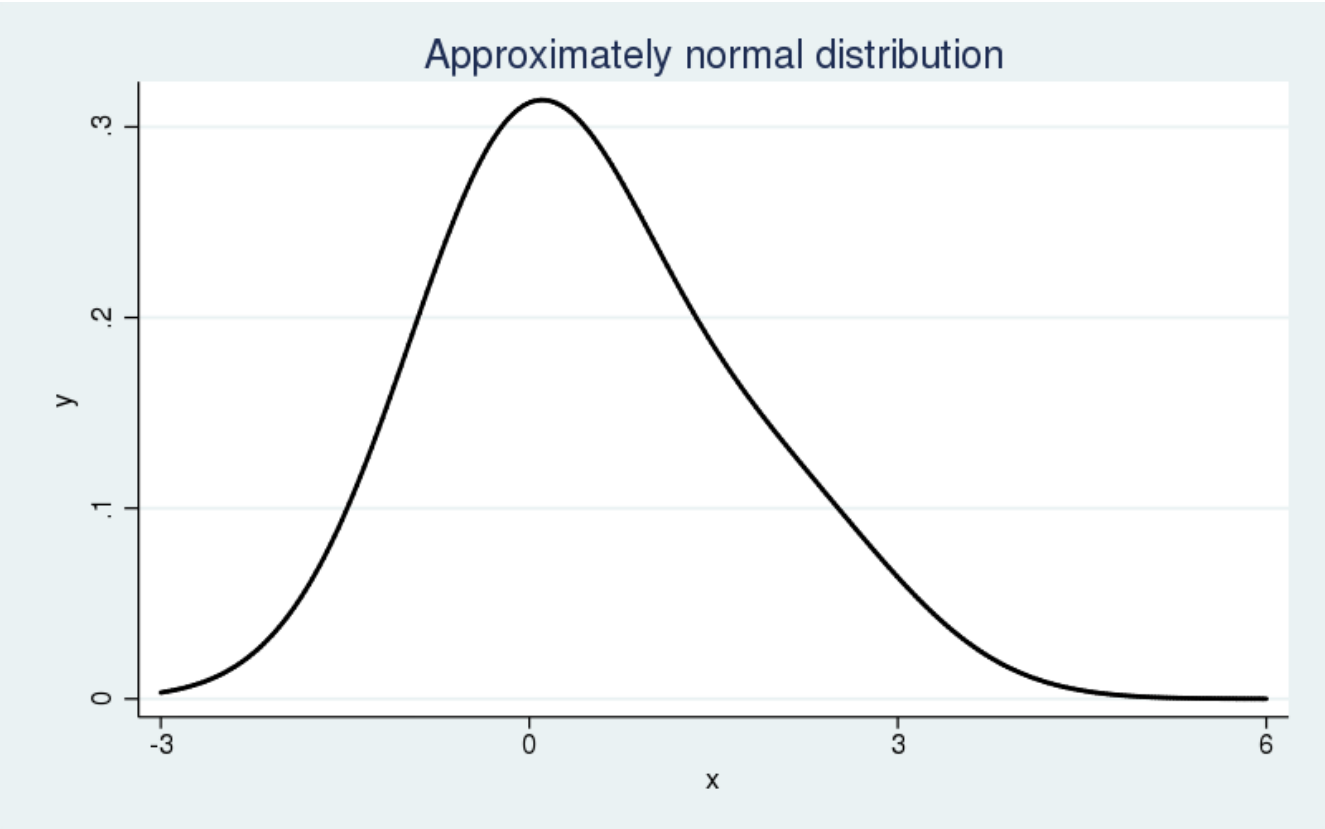
Group variable is not observed – apples from different orchards

Group variable is inherently unobservable – genetic predisposition to eating disorders

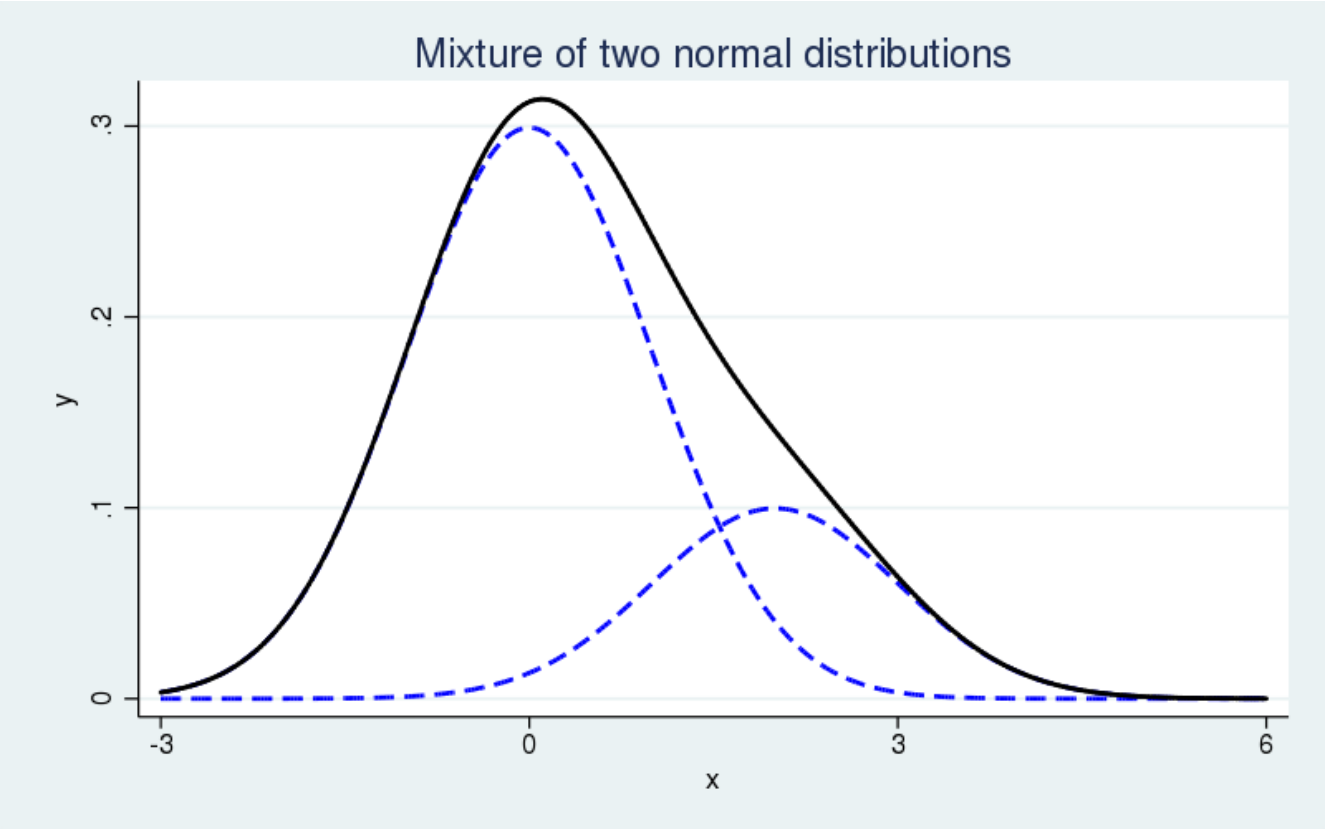
Group variable is too costly or impractical to measure – sex of goldfish

Group variable is misrepresented – dishonest reporting of illegal drug use

Let's take a step back from regression models for a few minutes and think about a single variable. We might have a variable, call it x, that has this distribution:



But perhaps this is actually a mixture of two normals.



How might we model this?

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

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 π_i is the probability for the ith class, and

$f_i(\cdot)$ is the conditional probability density function (pdf) for the observed response in the ith class model.

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

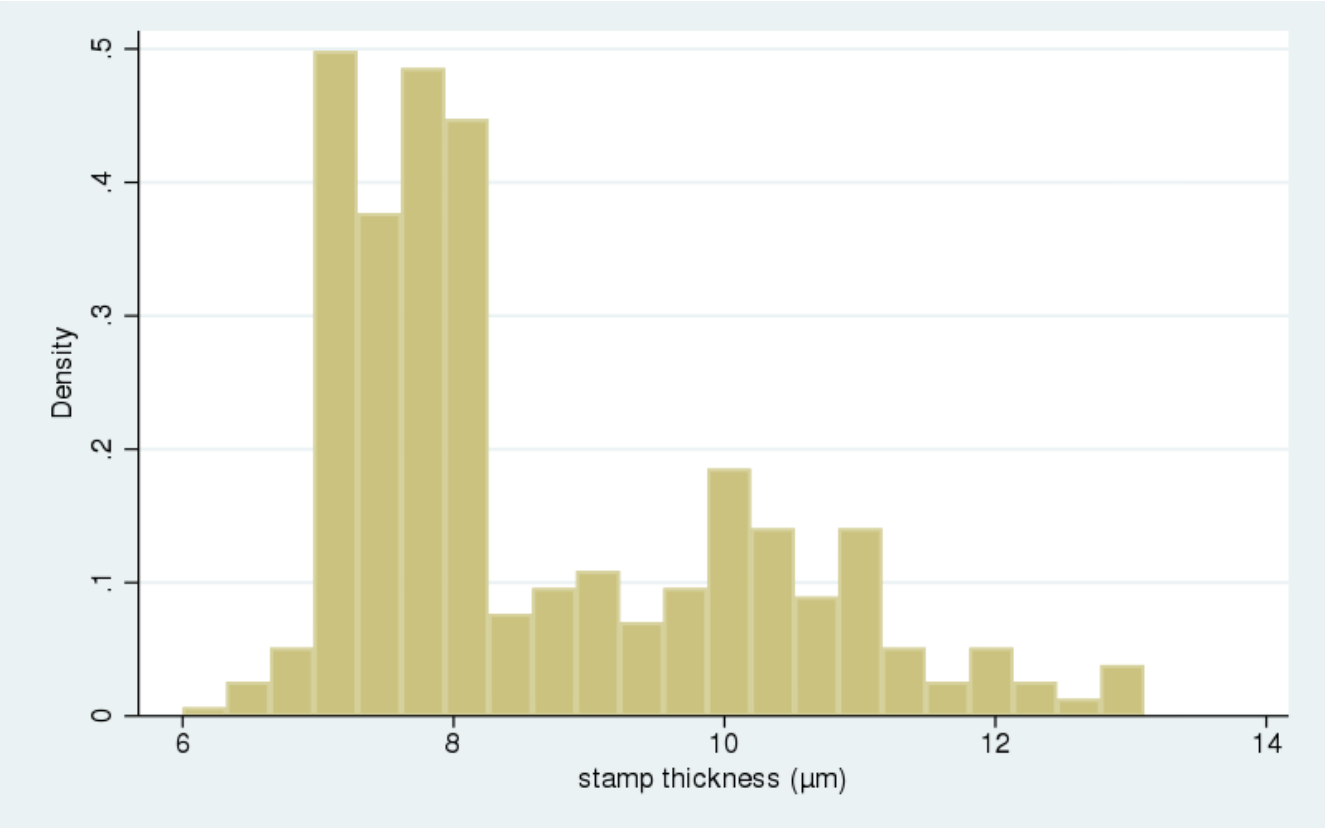
$$\pi_i = \frac{\exp(\gamma_i)}{\sum_{j=1}^{g} \exp(\gamma_j)}$$

where γ_i is the linear prediction for the ith latent class.

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

Example: Postal stamp thickness

```
. 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 "linear regression" or, in this case, "normal distribution"

Finite mixture model Number of obs = 485
Log likelihood = -748.75749

	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
1.Class	(base outcome)					
2.Class						
_cons	-.4498027	.124093	-3.62	0.000	-.6930205	-.2065848

Class : 1
Response : thick
Model : regress

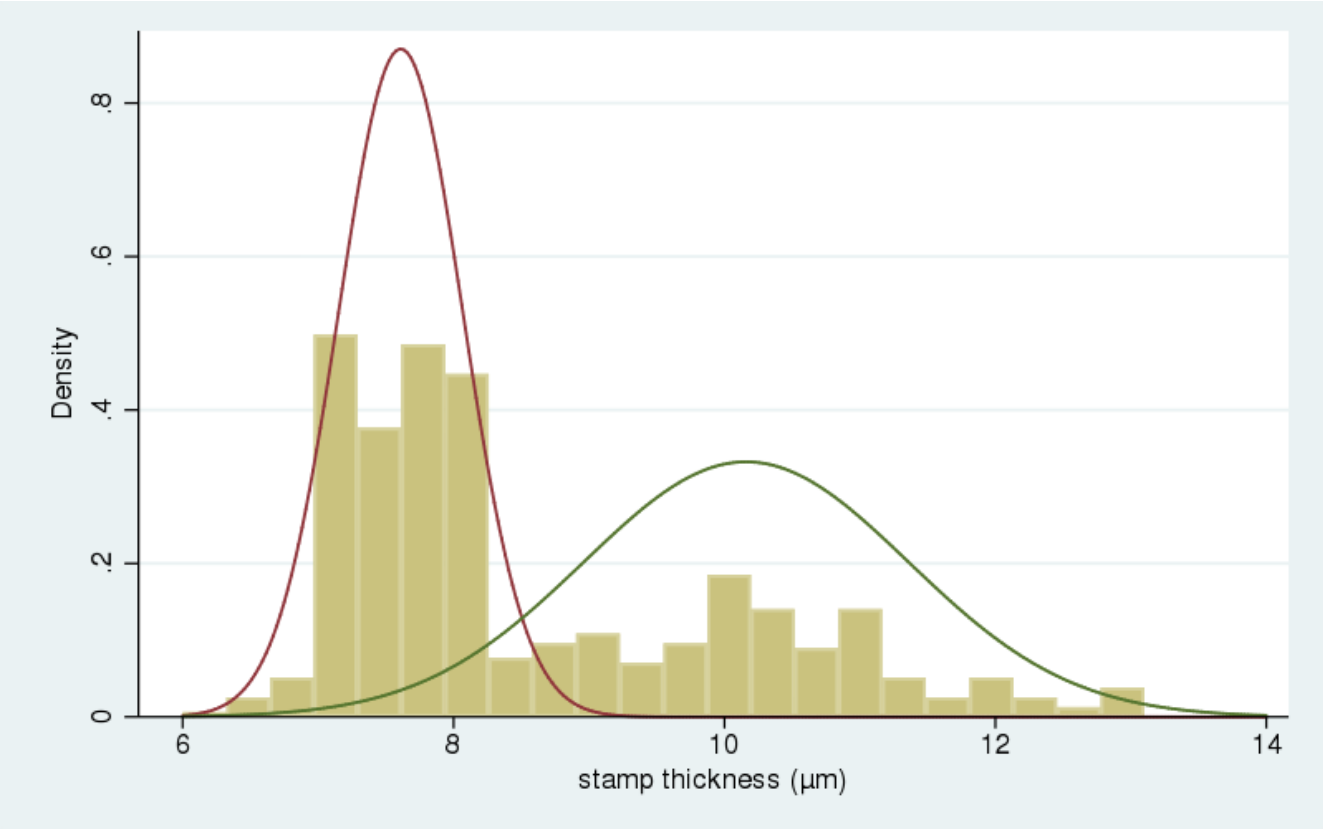
	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
thick						
_cons	7.609076	.0297275	255.96	0.000	7.550811	7.667341
var(e.thick)	.206297	.022201			.1670665	.2547395

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

Here are the two estimated distributions (without the mixing weights):

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



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 prob*, classposteriorpr

. describe prob1 prob2

              storage   display    value
variable name   type    format     label      variable label
------------------------------------------------------------------------------
prob1           float   %9.0g                 Predicted posterior probability (1.Class)
prob2           float   %9.0g                 Predicted posterior probability (2.Class)

. list thick prob1 prob2 ...
```

	thick	prob1	prob2
20.	8.3	.8123124	.1876876
21.	8.4	.7271801	.2728199
22.	8.5	.6116095	.3883905
23.	8.6	.4715973	.5284027
24.	8.7	.3267154	.6732846

```
. summarize prob1 prob2
```

Variable	Obs	Mean	Std. Dev.	Min	Max
prob1	485	.6105923	.4519458	1.53e-30	.9829751
prob2	485	.3894077	.4519458	.0170249	1

If you do not care about observation-level probabilities and are interested only in class probabilities, use **estat lcprob**:

```
. 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

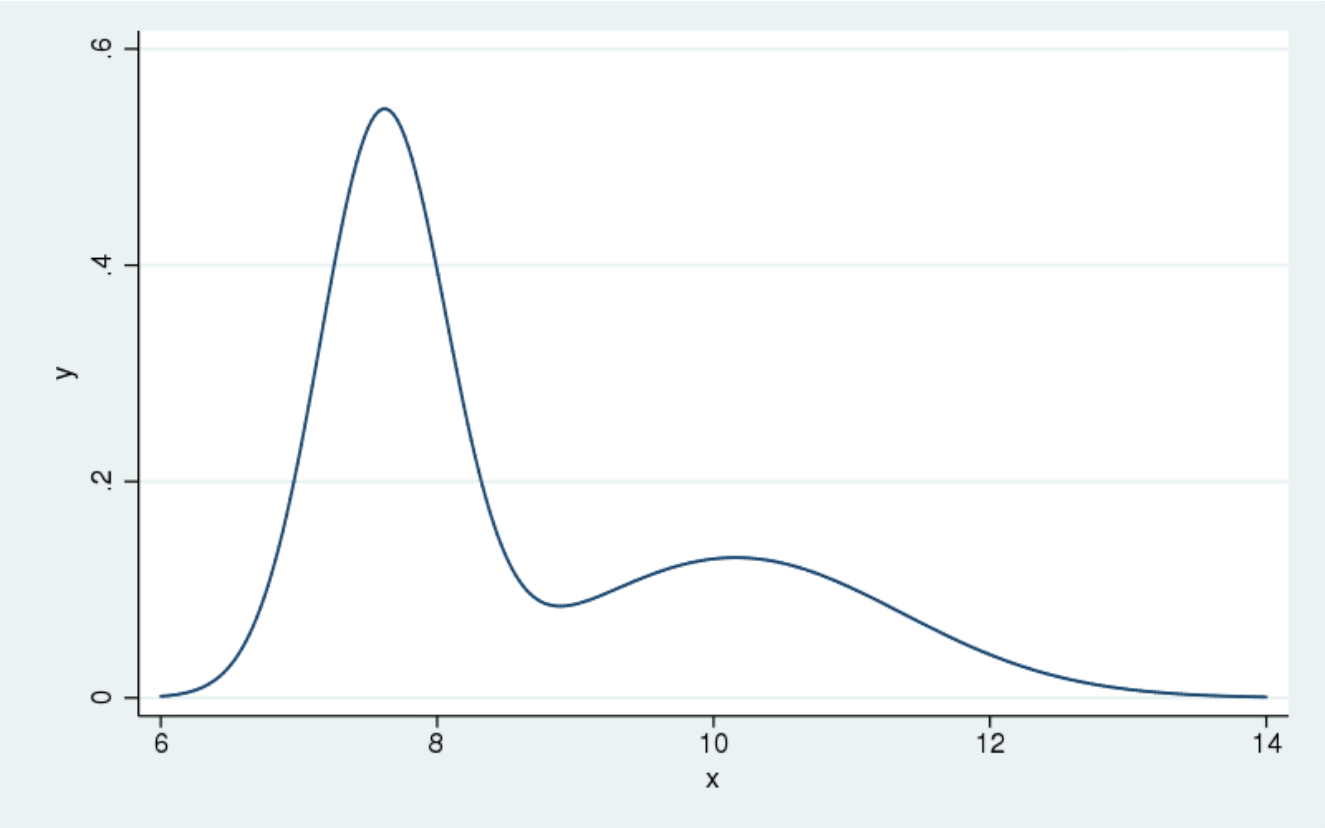
Whichever way you calculate class probabilities, this is our estimated mixture density:

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

We can graph this density in several different ways...

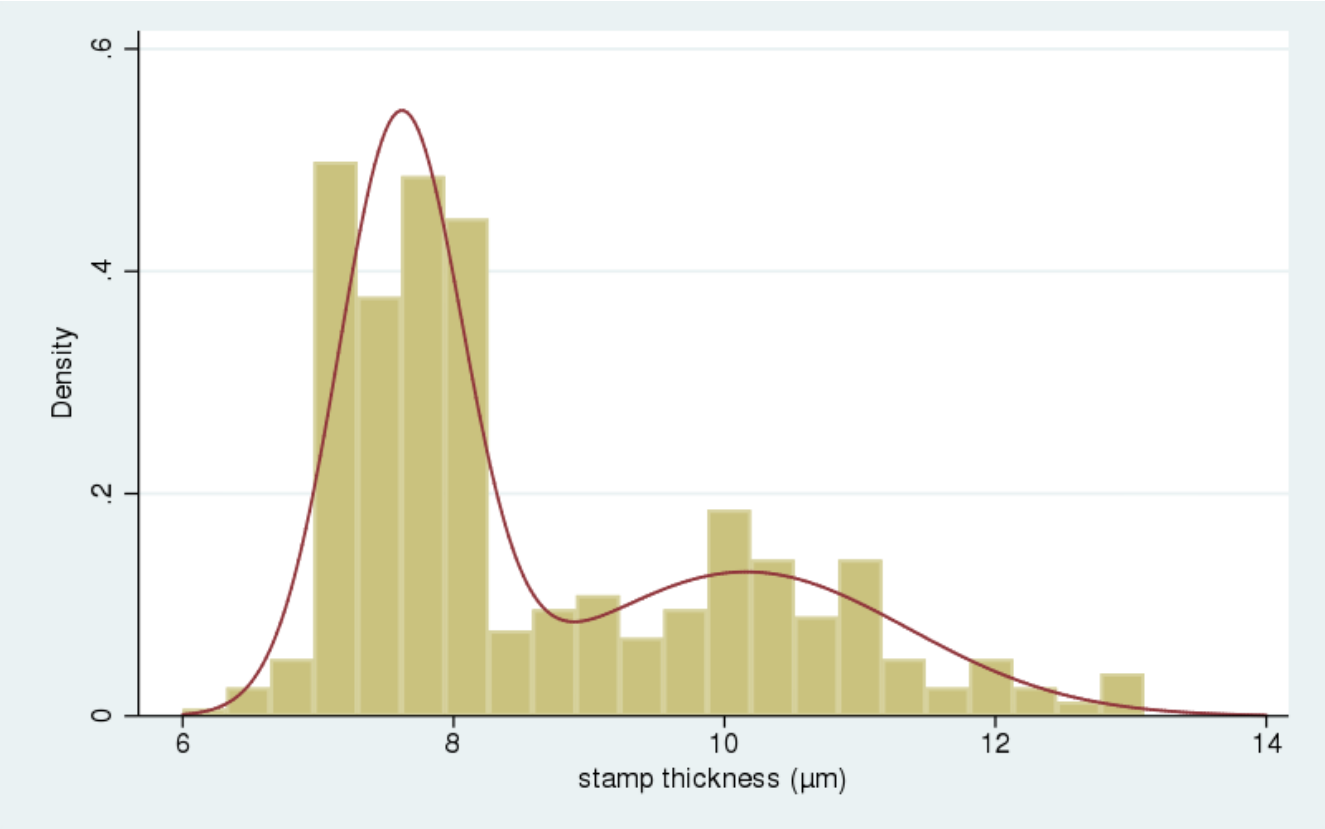
Plotting the mixture density (1)

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



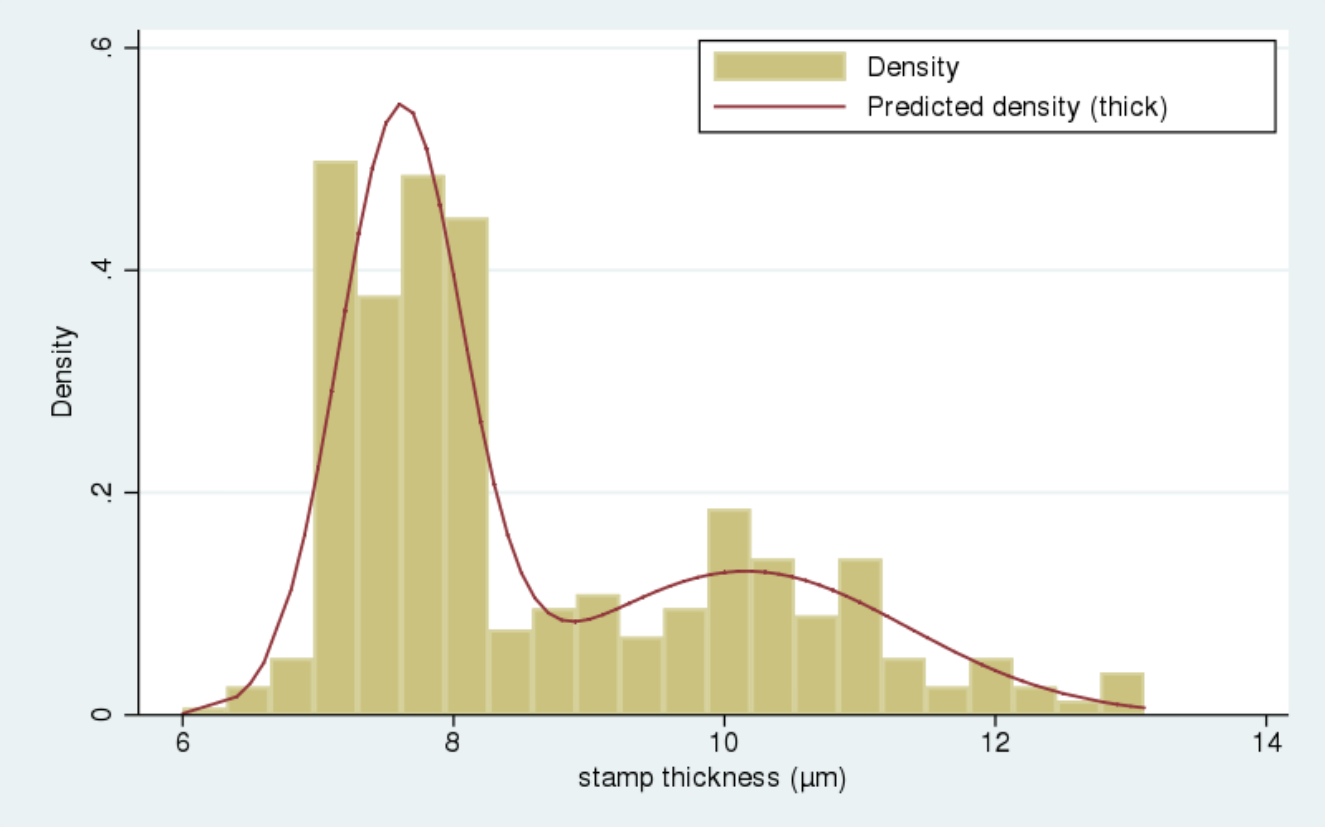
Plotting the mixture density (2)

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



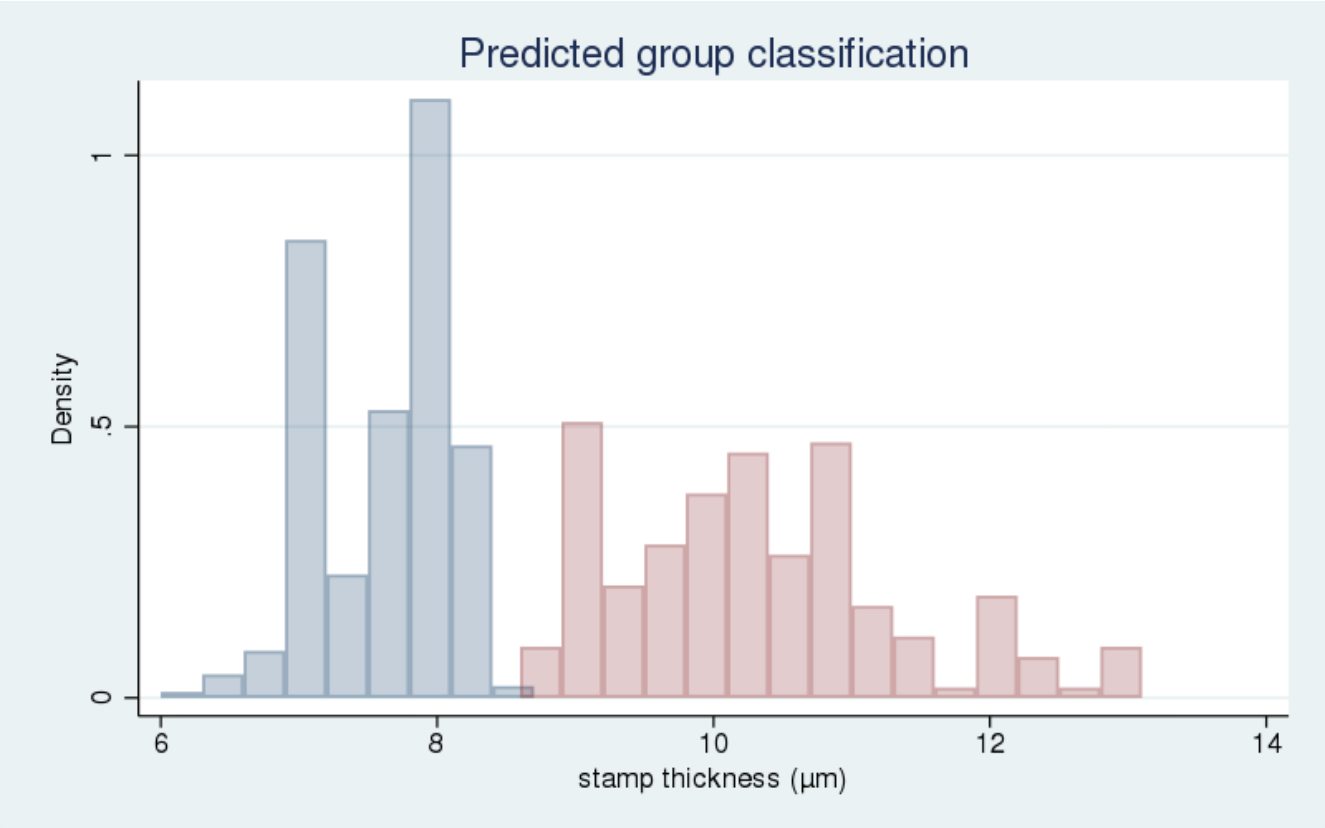
Plotting the mixture density (3)

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



We can also classify the stamps into the two groups.

```
. generate thin_paper = prob1 > .5
. twoway (histogram thick if thin_paper == 0 ...)  ///
         (histogram thick if thin_paper == 1 ...)
```



Now suppose we have regression models that differ across unobserved groups.

In this example, we will fit a mixture of two linear regression models.

```
. use chol, clear
(Artificial cholesterol data)

. describe
```

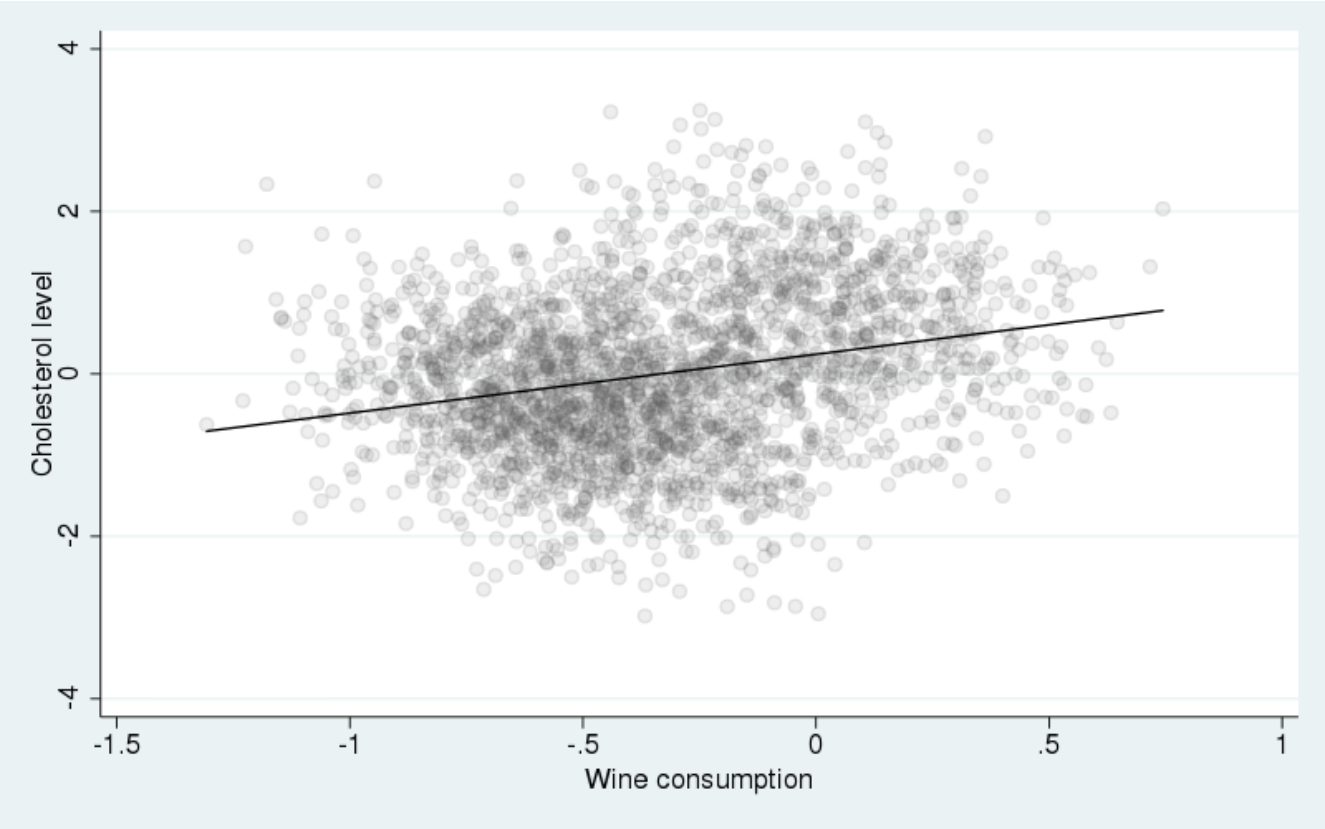
variable name	storage type	display format	value label	variable label
chol	float	%9.0g		Standardized cholesterol level
wine	float	%9.0g		Mean-centered monthly wine consumption

```
. 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 ...) ...
```



Suppose there are two types of individuals in the population: those with a "high cholesterol" gene and those without.

We proxy the presence of this gene by a respondent's parents' history.

```
. 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

```
. fmm 2, lcprob(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

	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

	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

Latent class means

```
. predict xb*
(option mu assumed)

. summarize xb1 xb2
```

Variable	Obs	Mean	Std. Dev.	Min	Max
xb1	2,500	-.5123399	.2395686	-1.249959	.1553361
xb2	2,500	.9938833	.1678007	.477234	1.461543

```
. estat lcmean

Latent class marginal means                     Number of obs     =      2,500
```

	Margin	Delta-method Std. Err.	z	P>|z|	[95% Conf.	Interval]
1						
chol	-.5123399	.024033	-21.32	0.000	-.5594438	-.465236
2						
chol	.9938833	.0601744	16.52	0.000	.8759435	1.111823

Class posterior probabilities

```
. predict prob*, classposteriorpr

. summarize prob1 prob2
```

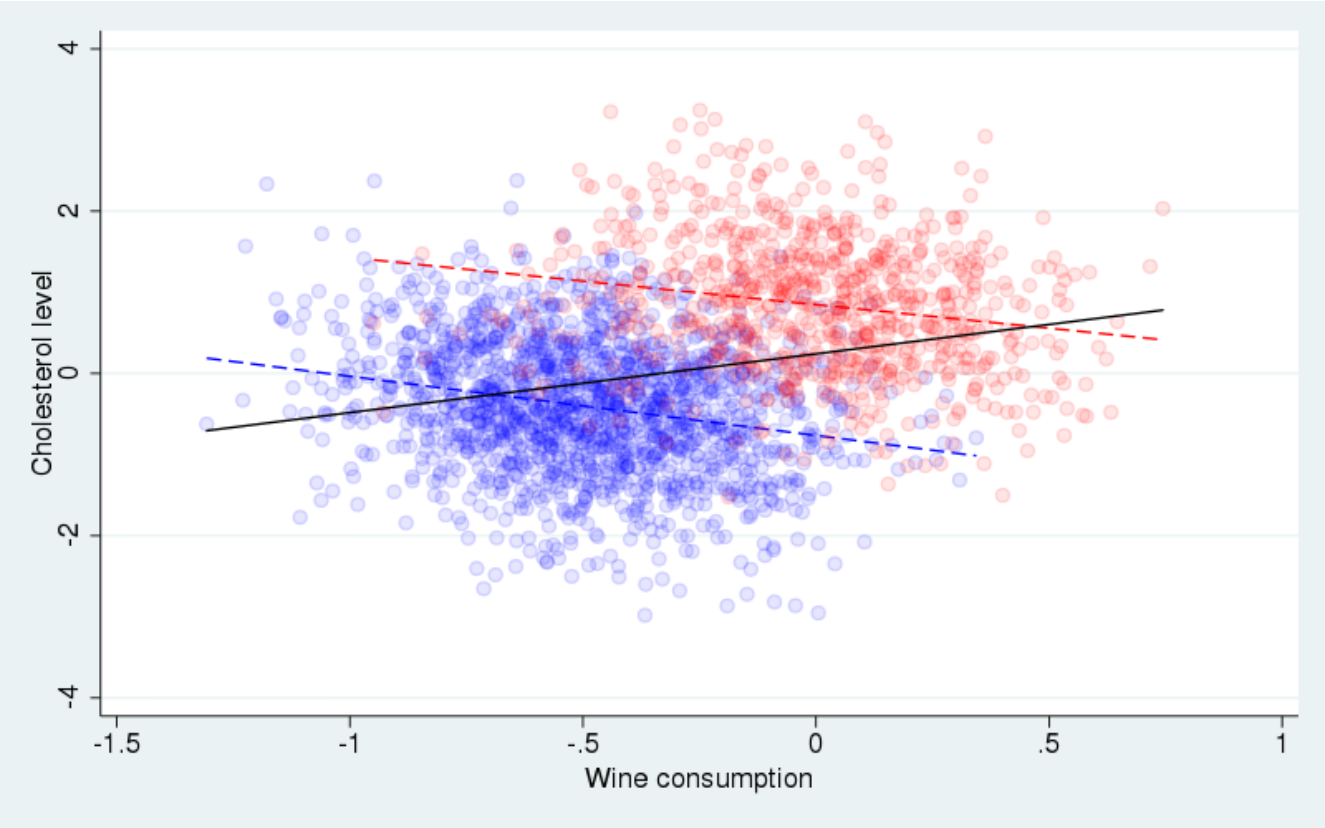
Variable	Obs	Mean	Std. Dev.	Min	Max
prob1	2,500	.6743291	.4538173	6.22e-09	1
prob2	2,500	.3256709	.4538173	4.24e-11	1

```
. estat lcprob

Latent class marginal probabilities             Number of obs     =      2,500
```

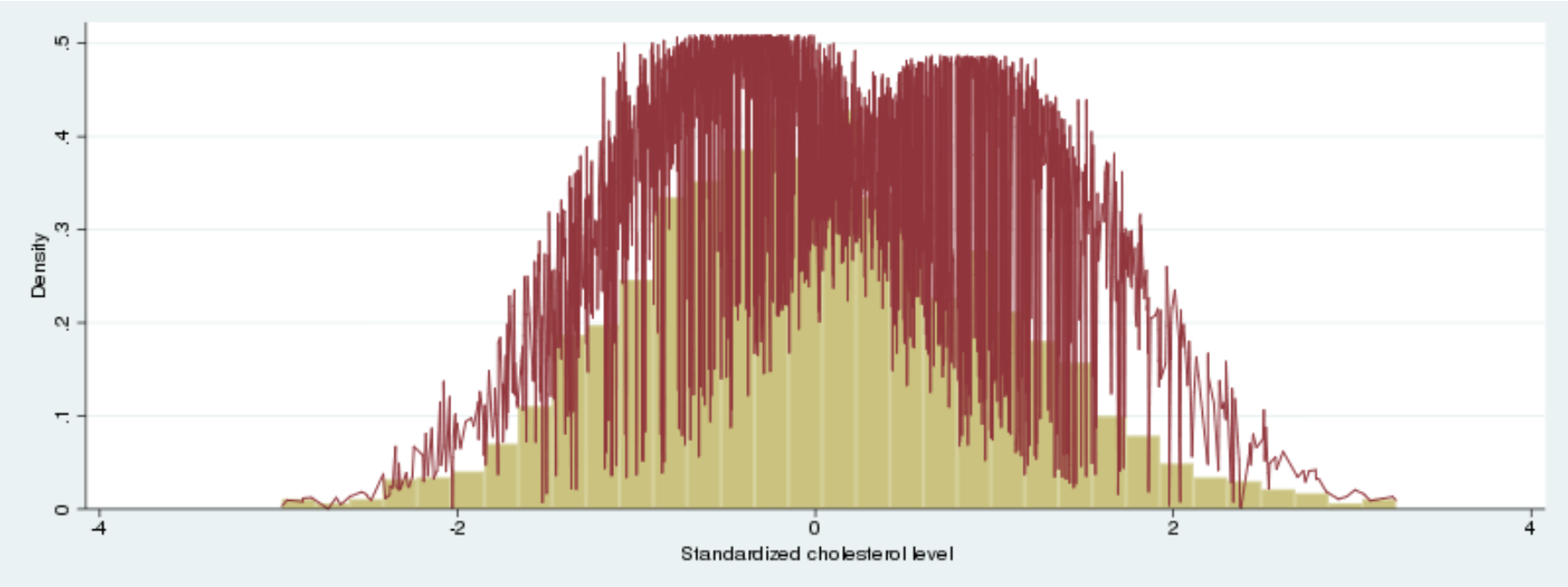
	Margin	Delta-method Std. Err.	[95% Conf.	Interval]
Class				
1	.6743291	.0055936	.6632719	.6851956
2	.3256709	.0055936	.3148044	.3367281

```
. gen hi_chol = prob2 > .5
. twoway (scatter chol wine if hi_chol == 0 ...) (lfit chol wine if hi_chol == 0 ...) ///
         (scatter chol wine if hi_chol == 1 ...) (lfit chol wine if hi_chol == 1 ...) ///
         (lfit chol wine ...) ...
```

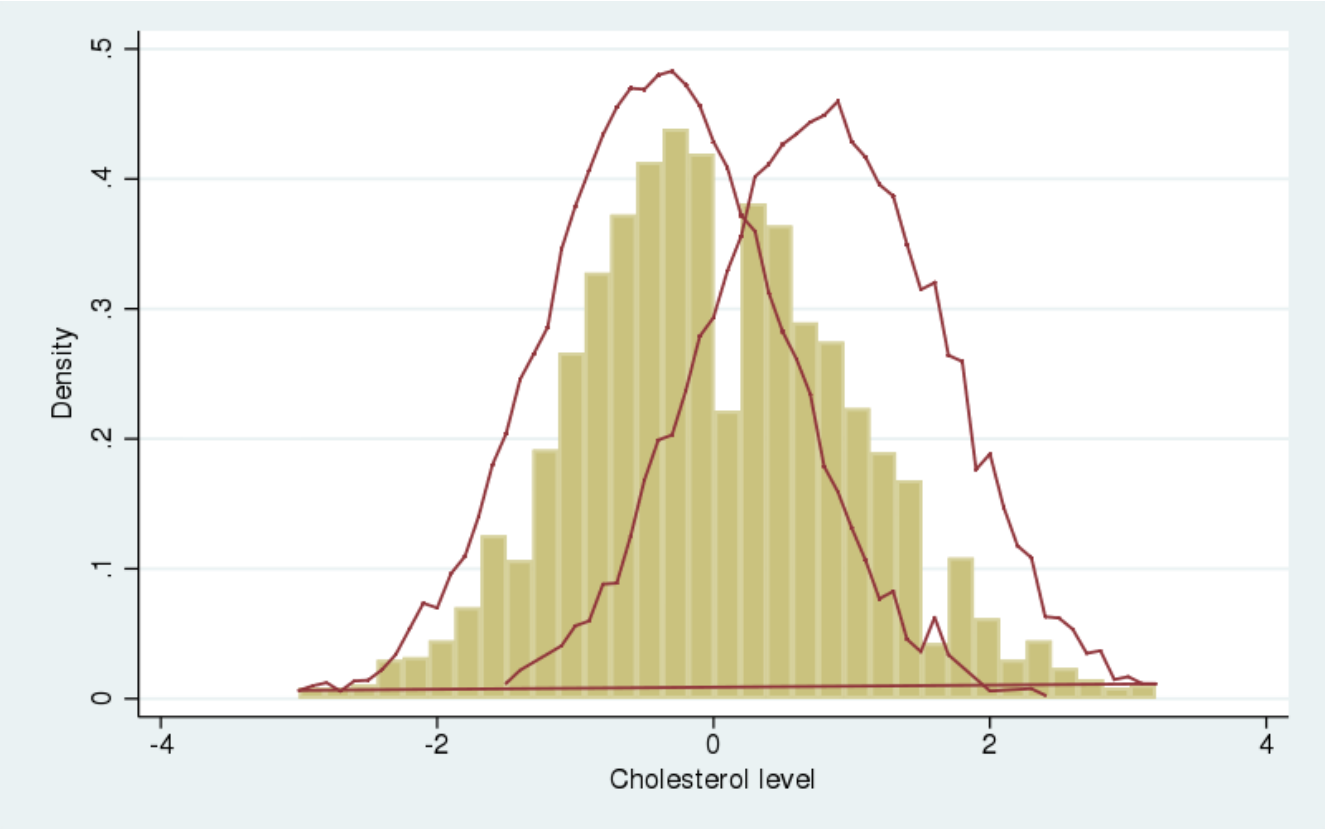


Some tips . . .

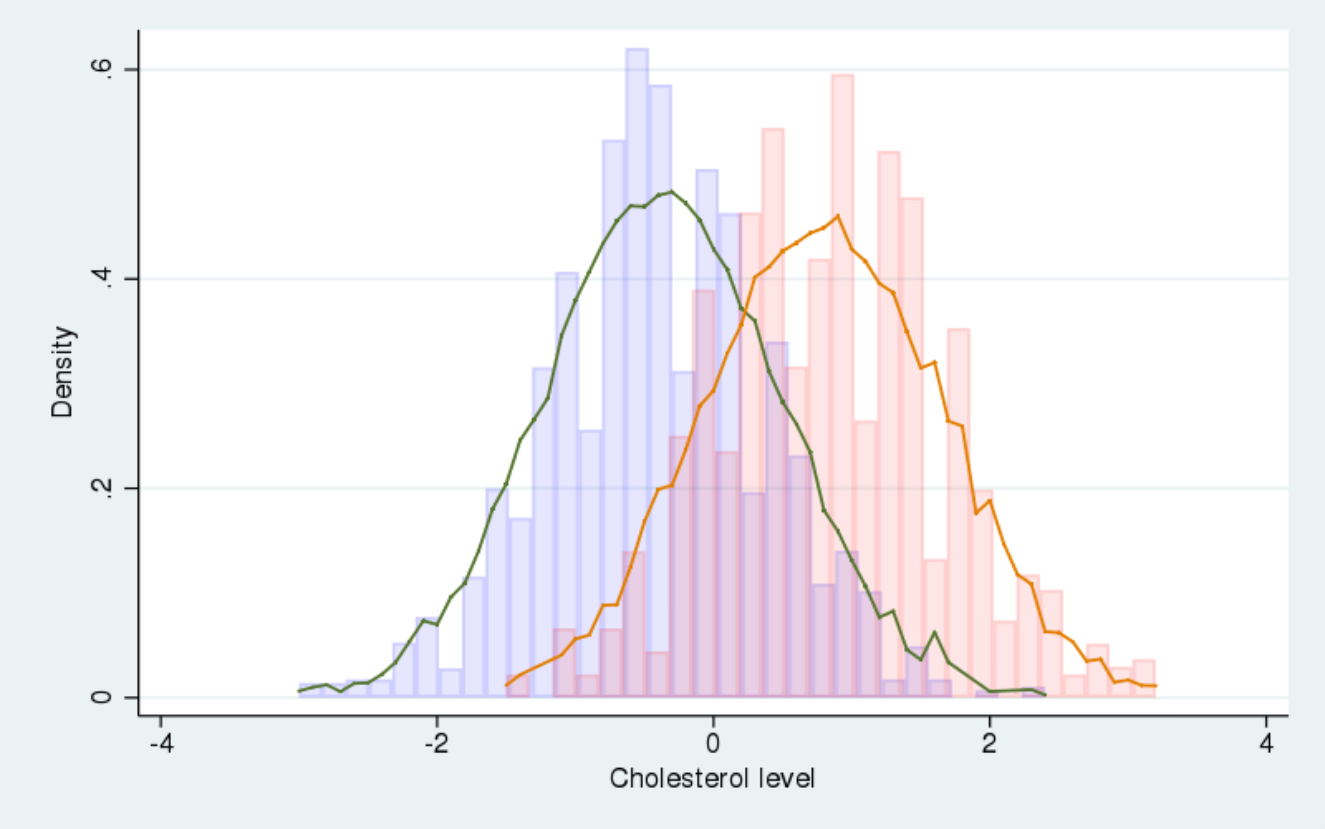
```
. predict den, density marginal
. sort chol
. histogram chol, addplot(line den chol) legend(off)
```



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



```
. twoway ///
      (histogram chol2 if hi_chol == 0, color(blue%10)) (line den2 chol2 if hi_chol == 0) ///
      (histogram chol2 if hi_chol == 1, color(red%10) ) (line den2 chol2 if hi_chol == 1) ...
```



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

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

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

Ordered categorical outcomes: **ologit** and **oprobit**

Unordered categorical 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**

For details, see [FMM] Finite mixture models using the fmm prefix

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 latent class membership using the **lcprob()** option.

Simple syntax

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

Equivalent hybrid syntax

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

Again, hybrid syntax is more flexible

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

Zero-inflated models

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
. summarize 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, lcprob(child camper))
```

```
Finite mixture model                            Number of obs     =        250
Log likelihood = -850.70142
```

	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

Latent class means and probabilities

```
. estat lcprob

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

	Delta-method Margin	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))
```

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

Beyond zero-inflated models

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 x1 x2) (pointmass y, value(1))

. fmm: (ologit y x1 x2) (pointmass y, value(2)) (pointmass y, value(4))

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

See the FMM manual for more examples:

[FMM] Mixture of linear regression models

[FMM] Mixture of Poisson regression models

[FMM] Mixture cure models for survival data

When **gsem** is preferable

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

```
fmm 2: regress chol wine = ...    // can't do!
```

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

A mixture of two normal distributions

fmm syntax

```
fmm 2: regress chol
```

gsem syntax

```
gsem (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)
```

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 might be able to guess how to use **gsem** 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)
```

See [SEM] example 54g for a worked example of how to fit a finite mixture model with more than one response variable.

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

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

gsem ///
        (1: count <-                    , family(pointmass 0))  ///
        (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)
```

When you have more than one latent class, you get interactions between the classes.

For example, to specify an equation for C = 2 and D = 3, we type:

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

For further details, see [SEM] intro 2 – Specifying generalized SEMs: Latent class analysis, two latent variables

Summary

Finite-mixture models are used to model unobserved heterogeneity

In Stata, you can use **fmm** to fit (mainly) univariate FMMs with a single latent class variable

In Stata, you can use **gsem** to fit univariate and multivariate FMMs with one or more latent classes

Use **estat lcmean** to obtain latent class means and **estat lcprob** to obtain latent class probabilities

Some useful links

FMM manual

[SEM] intro 2 – Specifying generalized SEMs: Latent class analysis

[SEM] intro 2 – Specifying generalized SEMs: Latent class analysis, class predictors

[SEM] intro 2 – Specifying generalized SEMs: Latent class analysis, two latent variables