

Inference for imputed latent classes using multiple imputation

Stas Kolenikov
NORC
Columbia, Missouri, USA
kolenikov-stas@norc.org

Abstract. I introduce a command to multiply impute latent classes following `gsem`, `lclass()` latent class analysis. This allows properly propagating uncertainty in class membership to downstream analysis that may characterize the demographic composition of the classes, or use the class as a predictor variable in statistical models.

Keywords: `st0001`, `postlca_class_predpute`, latent class analysis, multiple imputation

1 Latent class analysis

Latent class analysis (LCA) is a commonly used statistical and quantitative social science technique of modeling counts in high dimensional contingency tables, or tables of associations of categorical variables Hagenaars and McCutcheon (2002); McCutcheon (1987). LCA is a form of loglinear modeling, so let us explain that first. If the researcher has several categorical variables X_1, X_2, \dots, X_p with categories 1 through $m_j, j = 1, \dots, p$, at their disposal, and can produce counts $n_{k_1 k_2 \dots k_p}$ in a complete p -dimensional table, the first step could be modeling in main effects:

$$\mathbb{E} \log n_{k_1 k_2 \dots k_p} = \text{offset} + \sum_{j=1}^p \sum_{k=1}^{m_j} \beta_{jk_j} \quad (1)$$

with applicable identification constraints (such as the sum of the coefficients of a single variable is zero, or the coefficient for the first category of a variable is zero). Parameter estimates can be obtained by maximum likelihood, as equation (1) is a Poisson regression model. This model can be denoted as $X_1 + X_2 + \dots + X_p$ main effects model. The fit of the model is assessed by the Pearson χ^2 test comparing the expected vs. observed cell counts, or the likelihood ratio test against a saturated model where each cell has its own coefficient. If the model were to be found inadequate, the researcher can entertain adding interactions, e.g. the interaction of X_1 and X_2 would have $m_1 \times m_2$ terms for each pair of values of these variables, rather than $m_1 + m_2$ main effects:

$$\mathbb{E} \log n_{k_1 k_2 \dots k_p} = \text{offset} + \sum_{k_1=1}^{m_1} \sum_{k_2=1}^{m_2} \beta_{12, k_1 k_2} + \sum_{j=3}^p \sum_{k=1}^{m_j} \beta_{jk_j} \quad (2)$$

This model can be denoted as $X_1 \# X_2 + X_3 + \dots + X_p$.

In the loglinear model notation, the latent class models are models of the form $C \# (X_1 + X_2 + \dots + X_p)$. Categorical latent variable C is the latent class. The model is now a mixture of Poisson regressions, and maximum likelihood estimation additionally involves estimating the prevalence of each class of C .

Further extensions of latent class analysis may include:

1. Analysis with interactions of the observed variables;
2. Analysis with complex survey data (in which case estimation proceeds with `svy` prefix, and the counts are the weighted estimates of the population totals in cells);
3. Constrained analyses with structural zeroes or ones (e.g. that every member of class $C = 1$ must have the value $X_1 = 1$);
4. Constrained analyses where some variables have identical coefficients across classes.

1.1 Official Stata implementation

Official Stata `gsem`, `lclass()` implements the *main effects* LCA. The syntax is that of the SEM families, with the variables that the arrow points to interpreted as the outcome variables, and the latent class variable being the source of the arrow:

```
. webuse gsem_lcal
. gsem (accident play insurance stock <- ), logit lclass(C 2)
```

The goodness of fit test against the free tabulation counts is provided by `estat gof` (not available after the complex survey data analysis.)

As LCA is implemented through `gsem`, all the link functions and generalized linear model families are supported, extending the “mainstream” LCA.

1.2 Examples

LCA has found use in analyses of complicated economic concepts from survey data, and in assessment of measurement errors that arise in the process.

Biemer (2004) provided analysis of labor force classification status (employed, unemployed, and out of labor force) following changes in the survey instrument used in the Current Population Survey (CPS). The latent classes are the true LFS categories, and the observed variables are the corresponding survey measurements, demographics, and survey interview mode. The model is a variation of LCA that accounts for the survey methodology aspects of CPS: its panel nature (responses are collected over four consecutive months) and response mode (proxy reporting when a family member provides the survey responses rather than the target person.) He found that measurement of being employed and not being in labor force are highly accurate (98% and 97% accuracy)

while measurement of being unemployed is much less accurate (between 74% and 81% depending on the analysis year.) LCA allowed to further attribute the drop in accuracy of the unemployment status measurement to proxy reporting, and to the problems with measuring the employment status when the worker is laid off.

Kolenikov and Daley (2017) analyzed the latent classes of employees using the U.S. Department of Labor Worker Classification Survey. The observed variables were (composite) self-report of the employment status (are you an employee at your job; do you refer to your work as your business, your client, your job, etc.); tax status (the forms that the worker receives from their firm: W-2, 1099, K-1, etc.); behavioral control (functions the worker performs and the degree of control over these functions, such as direct reporting to somebody, schedule, permission to leave, etc.); and non-control composite (hired for fixed time or specific project). They found the best fitting model to contain three classes: employees-and-they-know-it (59%), nonemployees-and-they-know-it (24%), and confused (17%) who classify themselves as employees but their tax documentation is unclear, and other variables tend to place them into non-employee status.

1.3 Scope for this package

Researchers are often interested in describing the latent classes or using these classes in analysis as predictors or as moderators. The official [SEM] **gsem postestimation** commands provide limited possibilities, namely reporting of the means of the dependent variables by class via **estat lcmean**. For nearly all meaningful applications of LCA, this is insufficient.

The program distributed with the current package, **postlca_class_predpute**, provides a pathway for the appropriate statistical inference that would account for uncertainty in class prediction. This is achieved through the mechanics of multiple imputation (van Buuren 2018). The name is supposed to convey that

1. it is supposed to be run after LCA as a post-estimation command;
2. it predicts / imputes the latent classes.

2 The new command

Imputation of latent classes, a **gsem** postestimation command:

```
\begin{stsyntax}
  postlca\_class\_predpute,
  lcimpute(\varname)
  addm(\num)
  \optional{ seed(\num) }
\end{stsyntax}
```

lcimpute(*varname*) specifies the name of the latent class variable to be imputed. This option is required.

`addm(#)` specifies the number of imputations to be created. This option is required.

`seed(#)` specifies the random number seed.

3 Examples

3.1 Stata manual data set example

The LCA capabilities of Stata are exemplified in [SEM] **Example 50g**:

```
. frame change default
. cap frame gsem_lca1: clear
. cap frame drop gsem_lca1
. frame create gsem_lca1
. frame change gsem_lca1
.
. webuse gsem_lca1.dta, clear
(Latent class analysis)
. describe
Contains data from https://www.stata-press.com/data/r18/gsem_lca1.dta
Observations:      216      Latent class analysis
Variables:         4      17 Jan 2023 12:52
                        (_dta has notes)
```

Variable name	Storage type	Display format	Value label	Variable label
accident	byte	%9.0g		Would testify against friend in accident case
play	byte	%9.0g		Would give negative review of friend's play
insurance	byte	%9.0g		Would disclose health concerns to friend's insurance company
stock	byte	%9.0g		Would keep company secret from friend

Sorted by: accident play insurance stock

```
. gsem (accident play insurance stock <-), logit lclass(C 2)
(output omitted)
```

Generalized structural equation model Number of obs = 216
Log likelihood = -504.46767

	Coefficient	Std. err.	z	P> z	[95% conf. interval]	
1.C	(base outcome)					
2.C						
_cons	-.9482041	.2886333	-3.29	0.001	-1.513915	-.3824933

```
Class:    1
Response: accident
Family:   Bernoulli
Link:     Logit
Response: play
Family:   Bernoulli
Link:     Logit
```

```

Response: insurance
Family:   Bernoulli
Link:     Logit
Response: stock
Family:   Bernoulli
Link:     Logit

```

	Coefficient	Std. err.	z	P> z	[95% conf. interval]	
accident _cons	.9128742	.1974695	4.62	0.000	.5258411	1.299907
play _cons	-.7099072	.2249096	-3.16	0.002	-1.150722	-.2690926
insurance _cons	-.6014307	.2123096	-2.83	0.005	-1.01755	-.1853115
stock _cons	-1.880142	.3337665	-5.63	0.000	-2.534312	-1.225972

```

Class:      2
(output omitted)

```

	Coefficient	Std. err.	z	P> z	[95% conf. interval]	
accident _cons	4.983017	3.745987	1.33	0.183	-2.358982	12.32502
play _cons	2.747366	1.165853	2.36	0.018	.4623372	5.032395
insurance _cons	2.534582	.9644841	2.63	0.009	.6442279	4.424936
stock _cons	1.203416	.5361735	2.24	0.025	.1525356	2.254297

One of the official post-estimation commands available after `gsem`, `lclass()` is the computation of the class-specific means of the outcome variables:

```

. set rmsg on
r; t=0.00 14:45:54
. estat lcprob

```

Latent class marginal probabilities Number of obs = 216

	Delta-method		
	Margin	std. err.	[95% conf. interval]
C			
1	.7207539	.0580926	.5944743 .8196407
2	.2792461	.0580926	.1803593 .4055257

```

r; t=1.13 14:45:55
. estat lcmean

```

Latent class marginal means Number of obs = 216

		Delta-method		
		Margin	std. err.	[95% conf. interval]
1	accident	.7135879	.0403588	.6285126 .7858194
	play	.3296193	.0496984	.2403573 .4331299
	insurance	.3540164	.0485528	.2655049 .4538042
	stock	.1323726	.0383331	.0734875 .2268872
2	accident	.9931933	.0253243	.0863544 .9999956
	play	.9397644	.0659957	.6135685 .9935191
	insurance	.9265309	.0656538	.6557086 .9881667
	stock	.769132	.0952072	.5380601 .9050206

r; t=4.48 14:46:00

. set rmsg off

The mutiple imputation version of this estimation task could look as follows:

```
. set rmsg on
r; t=0.00 14:46:00
. postlca_class_predpute, lcimpute(lclass) addm(10) seed(12345)
(216 missing values generated)
(10 imputations added; M = 10)
r; t=0.05 14:46:00
. mi estimate : prop lclass
Multiple-imputation estimates      Imputations      =          10
Proportion estimation              Number of obs   =          216
                                   Average RVI       =          0.4594
                                   Largest FMI       =          0.3319
                                   Complete DF      =          215
DF adjustment: Small sample      DF: min       =          55.99
                                   avg              =          55.99
Within VCE type: Analytic        max            =          55.99
```

		Normal		
		Proportion	Std. err.	[95% conf. interval]
lclass	1	.7236111	.0367281	.6500355 .7971867
	2	.2763889	.0367281	.2028133 .3499645

r; t=2.01 14:46:02

. mi estimate : mean accident, over(lclass)

```
Multiple-imputation estimates      Imputations      =          10
Mean estimation                    Number of obs   =          216
                                   Average RVI       =          0.3882
                                   Largest FMI       =          0.4485
                                   Complete DF      =          215
DF adjustment: Small sample      DF: min       =          35.59
                                   avg              =          116.62
Within VCE type: Analytic        max            =          197.64
```

	Mean	Std. err.	[95% conf. interval]
--	------	-----------	----------------------

c.accident@lclass					
1	.7144964	.0369935	.6415438	.7874491	
2	.9934973	.0135709	.9659633	1.021031	

Note: Numbers of observations in e(_N) vary among imputations.
 r; t=2.40 14:46:04
 . set rmsg off

The name of the latent class variable (here, `lclass`) and the number of imputations are required. The seed is optional, but of course is strongly recommended for reproducibility of the results, as the underlying data are randomly simulated. The multiple imputation version is notably faster.

As one of many diagnostic outputs of MI, the increase in variances / standard errors due to imputations serves as an indication of how much of a problem would treating the singly imputed (e.g. modal probability) latent classes would have been. In the above output, the fraction of missing data (FMI) is 33% to 40%, and the relative variance increase (RVI) is the similar range from 39% to 45%. This means that the analysis with the deterministic (modal) imputation of the classes would have had standard errors that are about 20% too small.

```
. webuse gsem_lcal1.dta, clear
(Latent class analysis)
. quietly gsem (accident play insurance stock <-), logit lclass(C 2)
. predict post_1, class(1) classposterior
. gen byte lclass_modal = 2 - (post_1 > 0.50)
. mean post_1 lclass_modal
```

Mean estimation		Number of obs = 216		
	Mean	Std. err.	[95% conf. interval]	
post_1	.7207539	.0257112	.6700756	.7714321
lclass_modal	1.328704	.0320361	1.265559	1.391849

```
. mean accident, over(lclass_modal)
```

Mean estimation		Number of obs = 216		
	Mean	Std. err.	[95% conf. interval]	
c.accident@lclass_modal				
1	.6896552	.0385529	.6136651	.7656452
2		1	0	.

3.2 NHANES complex survey data example

In many important and realistic applications of LCA, including the case that necessitated the development of this package, the data come from complex survey designs that require setting the data up for the appropriate survey-design adjusted analyses. See

[SVY] `svyset`, [MI] `mi svyset`, and Kolenikov and Pitblado (2014).

The standard data set for the [SVY] commands is an extract from the National Health and Nutrition Examination Survey, Round Two (NHANES II) data. I will use a handful of binary health outcomes and one ordinal outcome to demonstrate LCA; the ordinal outcome is arguably an extension that is not quite well covered in the “classical” social science LCA.

```
. frame change default
. cap frame nhanes2: clear
. cap frame drop nhanes2
. frame create nhanes2
. frame change nhanes2
.
. webuse nhanes2.dta, clear
. svyset
Sampling weights: finalwtg
                  VCE: linearized
                  Single unit: missing
                  Strata 1: strata
Sampling unit 1: psu
                  FPC 1: <zero>
. svy , subpop(if hlthstat<8) : ///
>       gsem ///
>       (heartatk diabetes highbp <-, logit) ///
>       (hlthstat <-, ologit) ///
>       , lclass(C 2) nolog startvalues(randomid, draws(5) seed(101))
(running gsem on estimation sample)
Survey: Generalized structural equation model
Number of strata = 31
Number of PSUs   = 62
Number of obs    = 10,351
Population size  = 117,157,513
Subpop. no. obs  = 10,335
Subpop. size     = 116,997,257
Design df        = 31
```

	Linearized				
	Coefficient	std. err.	t	P> t	[95% conf. interval]
1.C	(base outcome)				
2.C					
_cons	1.330043	.1259401	10.56	0.000	1.073186 1.586899

```
Class:    1
Response: heartatk
Family:   Bernoulli
Link:     Logit
Response: diabetes
Family:   Bernoulli
Link:     Logit
Response: highbp
Family:   Bernoulli
Link:     Logit
```

Number of obs = 10,335

Number of obs = 10,335

Number of obs = 10,335


```

Response: hlthstat
Family:   Ordinal
Link:     Logit

```

Number of obs = 10,335

	Linearized					
	Coefficient	std. err.	t	P> t	[95% conf. interval]	
heartatk _cons	-1.874967	.1150791	-16.29	0.000	-2.109672	-1.640261
diabetes _cons	-1.785271	.0805057	-22.18	0.000	-1.949463	-1.621078
highbp _cons	.4244921	.076861	5.52	0.000	.2677332	.5812511
/hlthstat						
cut1	-3.659014	.8903346			-5.474863	-1.843165
cut2	-2.272516	.4402984			-3.17051	-1.374521
cut3	-.2566588	.2032721			-.671235	.1579173
cut4	1.229244	.1951641			.8312038	1.627283

Class: 2

```

Response: heartatk
Family:   Bernoulli
Link:     Logit

```

Number of obs = 10,335

```

Response: diabetes
Family:   Bernoulli
Link:     Logit

```

Number of obs = 10,335

```

Response: highbp
Family:   Bernoulli
Link:     Logit

```

Number of obs = 10,335

```

Response: hlthstat
Family:   Ordinal
Link:     Logit

```

Number of obs = 10,335

	Linearized					
	Coefficient	std. err.	t	P> t	[95% conf. interval]	
heartatk _cons	-6.081307	.6280801	-9.68	0.000	-7.362285	-4.800329
diabetes _cons	-5.223215	.6044468	-8.64	0.000	-6.455993	-3.990438
highbp _cons	-.8166105	.0750027	-10.89	0.000	-.9695795	-.6636415
/hlthstat						
cut1	-.657824	.0483113			-.7563555	-.5592926
cut2	.7123144	.0649814			.5797839	.8448448
cut3	2.647239	.1192958			2.403934	2.890544
cut4	24.64389	14.1421			-4.199113	53.48689

```

. set rmsg on
r; t=0.00 14:46:16
. estat lcprob

```

Latent class marginal probabilities

Number of strata = 31
Number of PSUs = 62

Number of obs = 10,351
Population size = 117,157,513
Design df = 31

	Delta-method			
	Margin	std. err.	[95% conf. interval]	
C				
1	.2091523	.0208315	.1698206	.2547976
2	.7908477	.0208315	.7452024	.8301794

r; t=10.03 14:46:26

. estat lcmean

Latent class marginal means

Number of strata = 31
Number of PSUs = 62

Number of obs = 10,351
Population size = 117,157,513
Design df = 31

	Delta-method			
	Margin	std. err.	[95% conf. interval]	
1				
heartatk	.1329681	.0132672	.1081603	.1624295
diabetes	.1436535	.0099036	.1246119	.1650562
highbp	.6045577	.018375	.5665363	.6413552
hlthstat				
Excellent	.0251111	.0217959	.0041733	.1366775
Very good	.0683138	.021455	.0355579	.127263
Good	.3427603	.0254834	.2928195	.3964437
Fair	.3375009	.0210993	.2958981	.3817814
Poor	.2263139	.0341724	.1642029	.3033906
2				
heartatk	.00228	.0014287	.0006343	.0081599
diabetes	.0053611	.0032231	.0015686	.0181559
highbp	.3064836	.0159419	.2749643	.3399221
hlthstat				
Excellent	.3412286	.01086	.319438	.3637112
Very good	.3296838	.0082507	.3130796	.346724
Good	.2629283	.0094265	.2441597	.2826002
Fair	.0661594	.0073704	.052623	.0828732
Poor	1.98e-11	2.81e-10	5.68e-24	.9857545

r; t=80.58 14:47:47

. set rmsg off

This analysis approximates breaking down the population into "generally healthy" and "unhealthy" groups, as e.g. the gradient of *hlthstat* variable between the classes shows. The official `gsem postestimation` commands take approximately forever to run (there is underlying `margins` implementation with iterations over the numeric derivatives step size used to compute the stadnard errors). There is an interaction of `svy` and `gsem` in that `svy` forces its own starting values that happen to be infe-

sible for LCA, hence the need to specify the initial random search. The use of the `postlca_class_predpute` command makes it possible to run the analysis much faster, and to conduct complementary analyses, e.g. analysis of the racial composition of the two classes.

```
. set rmsg on
r; t=0.00 14:47:47
. postlca_class_predpute, lcimpute(lclass) addm(10) seed(5678)
(10,351 missing values generated)
(10 imputations added; M = 10)
Sampling weights: finalwgt
                  VCE: linearized
                  Single unit: missing
                  Strata 1: strata
Sampling unit 1: psu
                  FPC 1: <zero>
r; t=0.22 14:47:47
. mi estimate : prop lclass
Multiple-imputation estimates      Imputations      =          10
Proportion estimation             Number of obs    =        10,351
                                  Average RVI       =         0.5318
                                  Largest FMI       =         0.3641
                                  Complete DF       =        10350
DF adjustment:   Small sample    DF:   min     =         73.85
                                  avg              =         73.85
Within VCE type:   Analytic      max              =         73.85
```

	Proportion	Std. err.	Normal [95% conf. interval]	
lclass				
1	.2675394	.0053851	.256809	.2782698
2	.7324606	.0053851	.7217302	.743191

```
r; t=1.53 14:47:49
. mi estimate : prop hlthstat if hlthstat < 8, over(lclass)
Multiple-imputation estimates      Imputations      =          10
Proportion estimation             Number of obs    =        10,335
                                  Average RVI       =          .
                                  Largest FMI       =          .
                                  Complete DF       =        10334
DF adjustment:   Small sample    DF:   min     =         36.31
                                  avg              =          .
Within VCE type:   Analytic      max              =          .
```

	Proportion	Std. err.	Normal [95% conf. interval]	
hlthstat@lclass				
Excellent 1	.0196036	.0034636	.0126501	.0265572
Excellent 2	.3104144	.0055292	.2995676	.3212613
Very good 1	.0550625	.0061217	.0426506	.0674743
Very good 2	.3217978	.0056545	.3106989	.3328966
Good 1	.2971359	.0106787	.2758813	.3183905
Good 2	.2795891	.0056354	.2685025	.2906756
Fair 1	.3635286	.0106978	.3423564	.3847008

Fair 2	.0881987	.003944	.0803607	.0960367
Poor 1	.2646694	.0089821	.2470257	.282313
Poor 2	0	(no observations)		

Note: Numbers of observations in e(_N) vary among imputations.
r; t=3.01 14:47:52

. mi estimate : prop race, over(lclass)

Multiple-imputation estimates	Imputations	=	10
Proportion estimation	Number of obs	=	10,351
	Average RVI	=	0.4677
	Largest FMI	=	0.4671
	Complete DF	=	10350
DF adjustment: Small sample	DF: min	=	45.27
	avg	=	119.43
Within VCE type: Analytic	max	=	321.80

	Proportion	Std. err.	Normal [95% conf. interval]	
race@lclass				
White 1	.839782	.0093485	.8209563	.8586078
White 2	.8889046	.0042799	.8804196	.8973897
Black 1	.1434614	.0085447	.1263568	.160566
Black 2	.0908358	.0038489	.0832192	.0984524
Other 1	.0167566	.0031195	.0105146	.0229986
Other 2	.0202596	.0017697	.0167778	.0237413

Note: Numbers of observations in e(_N) vary among imputations.
r; t=2.98 14:47:55

. set rmsg off

3.3 Choosing the number of imputations

One “researcher’s degrees of freedom” aspect of this analysis is the number of imputations M that need to be created. What this number affects the most is the stability of the standard errors obtained through the multiple imputation process. This stability is internally assessed with estimated degrees of freedom associated with the variance estimate (Barnard and Rubin 1999). With $M = 10$ imputations, the smaller “poor health” class have about 50 degrees of freedom:

. mi estimate : prop race, over(lclass)			
Multiple-imputation estimates	Imputations	=	10
Proportion estimation	Number of obs	=	10,351
	Average RVI	=	0.4677
	Largest FMI	=	0.4671
	Complete DF	=	10350
DF adjustment: Small sample	DF: min	=	45.27
	avg	=	119.43
Within VCE type: Analytic	max	=	321.80

	Proportion	Std. err.	Normal [95% conf. interval]	
race@lclass				

White 1	.839782	.0093485	.8209563	.8586078
White 2	.8889046	.0042799	.8804196	.8973897
Black 1	.1434614	.0085447	.1263568	.160566
Black 2	.0908358	.0038489	.0832192	.0984524
Other 1	.0167566	.0031195	.0105146	.0229986
Other 2	.0202596	.0017697	.0167778	.0237413

Note: Numbers of observations in e(_N) vary among imputations.

. mi estimate, dftable

Multiple-imputation estimates	Imputations	=	10
Proportion estimation	Number of obs	=	10,351
	Average RVI	=	0.4677
	Largest FMI	=	0.4671
	Complete DF	=	10350
DF adjustment: Small sample	DF: min	=	45.27
	avg	=	119.43
Within VCE type: Analytic	max	=	321.80

	Proportion	Std. err.	df	Normal std. err.
race@lclass				
White 1	.839782	.0093485	45.3	34.13
White 2	.8889046	.0042799	106.2	18.59
Black 1	.1434614	.0085447	57.9	28.28
Black 2	.0908358	.0038489	126.3	16.62
Other 1	.0167566	.0031195	59.0	27.89
Other 2	.0202596	.0017697	321.8	9.38

Note: Numbers of observations in e(_N) vary among imputations.

The public use version of the NHANES II data uses the approximate design that has 62 PSU in 31 strata, resulting in 31 design degrees of freedom. The imputation degrees of freedom barely exceed that. Let us push the number of imputations up:

```
. webuse nhanes2.dta, clear
. qui svy , subpop(if hlthstat<8) : ///
> gsem ///
> (heartatk diabetes highbp <-, logit) ///
> (hlthstat <-, ologit) ///
> , lclass(C 2) nolog startvalues(randomid, draws(5) seed(101))
. postlca_class_predpute, lcimpute(lclass) addm(62) seed(9752)
(10,351 missing values generated)
(62 imputations added; M = 62)
Sampling weights: finalwgt
VCE: linearized
Single unit: missing
Strata 1: strata
Sampling unit 1: psu
FPC 1: <zero>
. mi estimate : prop race, over(lclass)
Multiple-imputation estimates Imputations = 62
Proportion estimation Number of obs = 10,351
Average RVI = 0.2483
Largest FMI = 0.2893
Complete DF = 10350
```

```

DF adjustment:  Small sample    DF:    min    =    671.70
                                     avg    =   1,786.62
Within VCE type:    Analytic      max    =   3,092.35

```

	Proportion	Std. err.	Normal [95% conf. interval]	
race@lclass				
White 1	.8369206	.0079313	.8213584	.8524827
White 2	.8900061	.0038498	.8824571	.897555
Black 1	.1455849	.007628	.1306162	.1605536
Black 2	.0900031	.0035541	.0830334	.0969728
Other 1	.0174946	.0029465	.0117091	.02328
Other 2	.0199908	.001709	.01664	.0233417

Note: Numbers of observations in e(_N) vary among imputations.

. mi estimate, dftable

```

Multiple-imputation estimates    Imputations    =        62
Proportion estimation           Number of obs   =       10,351
                                Average RVI       =        0.2483
                                Largest FMI        =        0.2893
                                Complete DF        =       10350
DF adjustment:  Small sample    DF:    min    =    671.70
                                     avg    =   1,786.62
Within VCE type:    Analytic      max    =   3,092.35

```

	Proportion	Std. err.	df	Normal std. err.
race@lclass				
White 1	.8369206	.0079313	1100.3	13.14
White 2	.8900061	.0038498	2639.5	7.08
Black 1	.1455849	.007628	1004.8	13.98
Black 2	.0900031	.0035541	2211.1	8.08
Other 1	.0174946	.0029465	671.7	18.45
Other 2	.0199908	.001709	3092.4	6.26

Note: Numbers of observations in e(_N) vary among imputations.

The MI degrees of freedom are now comfortably above 600. In many i.i.d. data situations, increasing the number of imputations to several dozens can often send the MI degrees of freedom to approximate infinity (the reported numbers are in hundreds of thousands). With complex survey designs that have limited degrees of freedom within each implicate, this may not materialize. Researchers are encouraged to adopt the workflow where, in parallel, they

1. start with a small number of imputations, like `addm(10)` in the example above, and develop the analysis code for all the substantive analyses, and
2. working with the key outcomes or analyses, experiment with several values of M to find a reasonable trade-off when degrees of freedom exceed the sample size for i.i.d. data, and/or exceed the design degrees of freedom for complex survey data by a factor of 3–5.

Then a chosen value of M can be used for all analyses in the paper.

Even a large number of replications may not protect the researcher from classes that may have structural zeroes. These produce zero standard errors and missing degrees of freedom and variance increase statistics:

```
.
. mi estimate , dftable : prop hlthstat if hlthstat < 8, over(lclass)
Multiple-imputation estimates      Imputations      =          62
Proportion estimation             Number of obs    =       10,335
                                   Average RVI       =          .
                                   Largest FMI        =          .
                                   Complete DF        =       10334
DF adjustment:  Small sample      DF:  min        =       234.37
                                   avg                =          .
Within VCE type:  Analytic        max            =          .
```

	Proportion	Std. err.	df	Normal std. err.
hlthstat@lclass				
Excellent 1	.0183993	.0033919	309.7	32.68
Excellent 2	.3111818	.0054853	6242.4	3.09
Very good 1	.0573944	.0062471	234.4	41.21
Very good 2	.3212476	.0056361	4001.1	5.03
Good 1	.2947403	.0104573	575.3	20.57
Good 2	.2804536	.0055871	2155.5	8.23
Fair 1	.3656239	.010418	1033.8	13.71
Fair 2	.087117	.00393	549.1	21.27
Poor 1	.2638421	.0087535	4603.6	4.40
Poor 2	0	(no observations)		

Note: Numbers of observations in e(_N) vary among imputations.

4 Simulation

So how much of a problem is the modal imputation of the latent classes? I ran a small simulation to investigate. Samples were taken from a model with the classes, five binary indicators and one additional continuous variable not used in the model as follows:

Variable	Class 1	Class 2	Class 3
Class probability	0.4	0.4	0.2
$\mathbb{P}[y_1 = 1 C]$	0.7	0.3	0.6
$\mathbb{P}[y_2 = 1 C]$	0.8	0.5	0.6
$\mathbb{P}[y_3 = 1 C]$	0.5	0.4	0.7
$\mathbb{P}[y_4 = 1 C]$	0.5	0.3	0.7
$\mathbb{P}[y_5 = 1 C]$	0.8	0.4	0.3
$y_6 \sim N(\mu_c, 1)$	1	$\sqrt{2} = 1.41$	$\sqrt{3} = 1.73$

The LCA model with outcomes y_1 through y_5 was estimated, and the classes were predicted using the posterior modal prediction, multiple imputation with $M = 5$ imputed data sets, and $M = 50$ imputed data sets. (To be precise, there was a single imputation with $M = 50$, but two versions were run: `mi estimate , imp(1/5)` for the limited application with $M = 5$, and without the `imp()` option for the full set of $M = 50$.) Inspecting the individual runs visually, a shorter set of imputations often results in insufficient detail, namely failing to capture realizations of (posterior) rare classes.

There were at least two complications with the simulation. First, the classes in any LCA model are only identified up to a permutation of the class labels. To wit, there are no distinguishable differences between estimates when say classes 1 and 2 are swapped in a model with 2+ classes. The likelihoods are the same, the point estimates are likely to be the same within numeric accuracy. In any particular run of the `gsem, lclass()` command, the classes depend first and foremost on the starting values. The help file `help gsem_estimation_options##startvalues()` outlines the possible options:

- In my own practical work, I typically use `startvalues(randomid)`, `draws(10)` or `draws(20)` to get this many random assignments of the starting classes, and having Stata run the EM algorithm to get some decent starting values for the gradient-based optimization methods.
- For the simulation purposes, you have to fix either the initial assignment of classes, or the the starting values of the estimates. The former is implementable with `startvalues(classid true_class)` since the latter is, of course, known. Strangely, in this simulation, this did not work out well as it resulted in convergence issues: it has been pushing the model into areas of the parameter space where the likelihood was too flat to climb out of.
- The resulting simulation specification I used was a combination of `from(b0)` `startvalues(jitter 0.1, draws(10))`. The value of the starting matrix `b0` was obtained from a sample of size 10^6 computed once outside of the simulation. Jitter was added to allow some exploration of the sample optima near that point.

The second complication of the simulation was that in some runs, the `mi estimate` calls with imputed classes were returning errors. This may be a risk of a third-party written imputation routine. The author will investigate this with Stata Corp developers as this article progresses.

. tabulate method			
method	Freq.	Percent	Cum.
modal	2,047	37.29	37.29
predpute_5	1,836	33.44	70.73
predpute_50	1,607	29.27	100.00
Total	5,490	100.00	

With these limitations in mind, here are the simulation results.

Class probabilities are biased for the multiple imputation class prediction methods (the population value for class 1 was 0.4.)

```
. mean class1pr if touse3, over(n_method)
```

Mean estimation		Number of obs = 4,797		
		Mean	Std. err.	[95% conf. interval]
c.class1pr@n_method				
	modal	.3938263	.004415	.3851708 .4024818
	predpute_5	.3646841	.0038655	.3571059 .3722624
	predpute_50	.364754	.0038643	.3571783 .3723298

The modal method exhibits greater variability in class probabilities than the imputation methods. The standard errors are severely biased down for all methods. Multiple imputation with more replicates has more stable standard errors.

```
. bysort method (rep): sum class1pr if touse3
```

-> method = modal					
Variable	Obs	Mean	Std. dev.	Min	Max
class1pr	1,583	.3938263	.1756604	.039	.942
-> method = predpute_5					
Variable	Obs	Mean	Std. dev.	Min	Max
class1pr	1,607	.3646841	.1549593	.0418	.8014
-> method = predpute_50					
Variable	Obs	Mean	Std. dev.	Min	Max
class1pr	1,607	.364754	.1549089	.04078	.80522

```
. mean class1se if touse3, over(n_method)
```

Mean estimation		Number of obs = 4,797		
		Mean	Std. err.	[95% conf. interval]
c.class1se@n_method				
	modal	.0143072	.0000464	.0142163 .0143981
	predpute_5	.0188952	.0000906	.0187177 .0190728
	predpute_50	.0184366	.0000579	.0183232 .0185501

Turning to the outcomes in the model, I picked two random summaries out of 15 (=5 outcomes \times 3 classes). For the outcome y_1 in class 2, estimates are biased away from the true value of 0.3 for the modal method:

```
. mean y1cl2pr if touse3 & y1cl2pr>0.01 & y1cl2pr<0.99, over(n_method)
```

Mean estimation		Number of obs = 4,503		
-----------------	--	-----------------------	--	--

	Mean	Std. err.	[95% conf. interval]	
c.y1cl2pr@n_method				
modal	.218858	.0027115	.2135422	.2241739
predpute_5	.2953609	.0020341	.291373	.2993489
predpute_50	.2954626	.002029	.2914848	.2994404

The same observations concerning the standard errors apply: the variability of the modal method estimates is greater than the multiple imputation estimates, the standard errors are biased for all methods, and the standard errors are more stable with the greater number of replicates.

```
. bysort method (rep): sum y1cl2pr if touse3 & y1cl2pr>0.01 & y1cl2pr<0.99
```

```
-> method = modal
```

Variable	Obs	Mean	Std. dev.	Min	Max
y1cl2pr	1,355	.218858	.0998111	.0234987	.702381

```
-> method = predpute_5
```

Variable	Obs	Mean	Std. dev.	Min	Max
y1cl2pr	1,574	.2953609	.0807021	.0179912	.5569573

```
-> method = predpute_50
```

Variable	Obs	Mean	Std. dev.	Min	Max
y1cl2pr	1,574	.2954626	.0804971	.0193931	.5578867

```
. mean y1cl2se if touse3 & y1cl2pr>0.01 & y1cl2pr<0.99, over(n_method)
```

```
Mean estimation
```

```
Number of obs = 4,503
```

	Mean	Std. err.	[95% conf. interval]	
c.y1cl2se@n_method				
modal	.0196673	.0001217	.0194288	.0199059
predpute_5	.0296163	.0002038	.0292168	.0300158
predpute_50	.029101	.0001752	.0287575	.0294444

Another outcome summarized, y_4 in class 1, has the target value of 0.5. Biases are in opposite directions for the modal method and multiple imputation methods.

```
. mean y4cl1pr if touse3 & y4cl1pr>0.01 & y4cl1pr<0.99, over(n_method)
```

```
Mean estimation
```

```
Number of obs = 4,687
```

	Mean	Std. err.	[95% conf. interval]	
c.y4cl1pr@n_method				
modal	.5134519	.0027162	.5081269	.5187769
predpute_5	.4861217	.0018979	.4824009	.4898424
predpute_50	.4862154	.001881	.4825277	.489903

Problems with the variability and its estimation with the standard errors persist.

```
. bysort method (rep): sum y4cl1pr if touse3 & y4cl1pr>0.01 & y4cl1pr<0.99
```

```
-> method = modal
```

Variable	Obs	Mean	Std. dev.	Min	Max
y4cl1pr	1,487	.5134519	.1047403	.0546875	.8440678

```
-> method = predpute_5
```

Variable	Obs	Mean	Std. dev.	Min	Max
y4cl1pr	1,600	.4861217	.0759155	.0521549	.9703288

```
-> method = predpute_50
```

Variable	Obs	Mean	Std. dev.	Min	Max
y4cl1pr	1,600	.4862154	.0752401	.0491738	.9807849

```
. mean y4cl1se if touse3 & y4cl1pr>0.01 & y4cl1pr<0.99, over(n_method)
```

```
Mean estimation
```

```
Number of obs = 4,687
```

	Mean	Std. err.	[95% conf. interval]	
c.y4cl1se@n_method				
modal	.0257628	.0001536	.0254616	.0260639
predpute_5	.0342909	.0002728	.0337561	.0348258
predpute_50	.0338453	.0002555	.0333444	.0343461

Moving now to investigate the variable outside the model, y_6 , we observe that the estimates are severely biased towards the pooled mean.

```
. mean y6cl1pr if touse3, over(n_method)
```

```
Mean estimation
```

```
Number of obs = 4,797
```

	Mean	Std. err.	[95% conf. interval]	
c.y6cl1pr@n_method				
modal	1.178873	.0019722	1.175007	1.18274
predpute_5	1.208982	.0015596	1.205925	1.21204
predpute_50	1.209956	.0014737	1.207067	1.212845

```
. mean y6cl3pr if touse3, over(n_method)
```

```
Mean estimation
```

```
Number of obs = 4,728
```

	Mean	Std. err.	[95% conf. interval]	
c.y6cl3pr@n_method				
modal	1.453111	.0034072	1.446431	1.45979
predpute_5	1.40193	.0021874	1.397642	1.406219
predpute_50	1.401776	.0020405	1.397776	1.405777

The modal method continues to exhibit greater variability than the imputation meth-

ods. For this outcome, however, the standard errors are much closer to their targets. The standard errors of the modal method (mean of 0.058 for class 1) underestimate the true Monte Carlo variability (mean of 0.078), while the multiple imputation standard errors are biased upwards (0.071 vs. the targets of 0.060). Multiple imputation with more replicates has more stable standard errors.

```
. bysort method (rep): sum y6cl1pr if touse3
```

```
-> method = modal
```

Variable	Obs	Mean	Std. dev.	Min	Max
y6cl1pr	1,583	1.178873	.0784673	.8461757	1.417481

```
-> method = predpute_5
```

Variable	Obs	Mean	Std. dev.	Min	Max
y6cl1pr	1,607	1.208982	.0625194	.897253	1.401546

```
-> method = predpute_50
```

Variable	Obs	Mean	Std. dev.	Min	Max
y6cl1pr	1,607	1.209956	.0590762	.9668165	1.388266

```
. mean y6cl1se if touse3, over(n_method)
```

```
Mean estimation
```

```
Number of obs = 4,797
```

	Mean	Std. err.	[95% conf. interval]	
c.y6cl1se@n_method				
modal	.0575567	.0004523	.05667	.0584435
predpute_5	.0723921	.0006426	.0711323	.073652
predpute_50	.0712562	.0005766	.0701258	.0723867

Same observations apply to the standard errors for class 3.

```
. bysort method (rep): sum y6cl3pr if touse3
```

```
-> method = modal
```

Variable	Obs	Mean	Std. dev.	Min	Max
y6cl3pr	1,514	1.453111	.132573	.9581373	2.053384

```
-> method = predpute_5
```

Variable	Obs	Mean	Std. dev.	Min	Max
y6cl3pr	1,607	1.40193	.0876854	1.136963	1.883443

```
-> method = predpute_50
```

Variable	Obs	Mean	Std. dev.	Min	Max
y6cl3pr	1,607	1.401776	.0817985	1.139892	1.755444

```
. mean y6cl3se if touse3, over(n_method)
```

Mean estimation		Number of obs = 4,728		
		Mean	Std. err.	[95% conf. interval]
c.y6cl3se@n_method				
	modal	.0923276	.0012236	.0899288 .0947263
	predpute_5	.1099187	.0013831	.1072071 .1126303
	predpute_50	.1078059	.0012745	.1053072 .1103046

Overall, it seems like the problem of the class indeterminacy (permutation of classes) has still been damaging the simulation. Further steps would need to be taken to better match the estimated classes with their population counterparts.

5 References

- Barnard, J., and D. B. Rubin. 1999. Small-Sample Degrees of Freedom with Multiple Imputation. *Biometrika* 86: 948–955.
- Biemer, P. P. 2004. An Analysis of Classification Error for the Revised Current Population Survey Employment Questions. *Survey Methodology* 30: 127–140.
- Hagenaars, J. A., and A. L. McCutcheon. 2002. *Applied latent class analysis*. Cambridge University Press.
- Kolenikov, S., and K. Daley. 2017. Latent Class Analysis of Worker Knowledge of Their Employment Status. 3898–3906. Alexandria, VA, USA. Available from www.asasrms.org/Proceedings/y2017/files/594102.pdf.
- Kolenikov, S., and J. Pitblado. 2014. Analysis of complex health survey data. In *Handbook of Health Survey Methods*, ed. T. P. Johnson, chap. 29. Hoboken, NJ: Wiley.
- McCutcheon, A. L. 1987. *Latent class analysis*. No. 64 in Quantitative Applications in Social Sciences—“The Little Green Book”, Sage.
- van Buuren, S. 2018. *Flexible Imputation of Missing Data*. 2nd ed. Chapman & Hall/CRC.

About the authors

Stas Kolenikov is Principal Statistician at NORC who has been using Stata and writing Stata programs for about 25 years. He had worked on economic welfare and inequality, spatiotemporal environmental statistics, mixture models, missing data, multiple imputation, structural equations with latent variables, resampling methods, complex sampling designs, survey weights, Bayesian mixed models, combining probability and non-probability samples, latent class analysis, and likely some other stuff, too.