The Stata Journal (2010)
10, Number 3, pp. 315–330

An introduction to maximum entropy and minimum cross-entropy estimation using Stata

Martin Wittenberg
University of Cape Town
School of Economics
Cape Town, South Africa
Martin.Wittenberg@uct.ac.za

Abstract. Maximum entropy and minimum cross-entropy estimation are applicable when faced with ill-posed estimation problems. I introduce a Stata command that estimates a probability distribution using a maximum entropy or minimum cross-entropy criterion. I show how this command can be used to calibrate survey data to various population totals.

Keywords: st0196, maxentropy, maximum entropy, minimum cross-entropy, survey calibration, sample weights

1 Ill-posed problems and the maximum entropy criterion

All too many situations involve more unknowns than data points. Standard forms of estimation are impossible when faced with such ill-posed problems (Mittelhammer, Judge, and Miller 2000). One approach that is applicable in these cases is estimation by maximizing an entropy measure (Golan, Judge, and Miller 1996). The purpose of this article is to introduce the concept and to show how to apply it using the new Stata command `maxentropy`. My discussion of the technique follows the treatment in Golan, Judge, and Miller (1996). Furthermore, I show how a maximum entropy approach can be used to calibrate survey data to various population totals. This approach is equivalent to the iterative raking procedure of Deming and Stephan (1940) or the multiplicative method implemented in the calibration on margins (CALMAR) algorithm (Deville and Särndal 1992; Deville, Särndal, and Sautory 1993).

The idea of maximum entropy estimation was motivated by Jaynes (1957, 621ff) in terms of the problem of finding the probability distribution $(p_1, p_2, \ldots, p_n)$ for the set of values $(x_1, x_2, \ldots, x_n)$, given only their expectation,

$$E\{f(x)\} = \sum_{i=1}^{n} p_i f(x_i)$$

For concreteness, we consider a die known to have $E(x) = 3.5$, where $\mathbf{x} = (1, 2, 3, 4, 5, 6)$, and we want to determine the associated probabilities. Clearly, there are infinitely many possible solutions, but the obvious one is $p_1 = p_2 = \cdots = p_6 = 1/6$. The obviousness is based on Laplace's principle of insufficient reason, which states that two events should be assigned equal probability unless there is a reason to think otherwise (Jaynes 1957, 622). This negative reason is not much help if, instead, we know that $E(x) = 4$. Jaynes's solution was to tackle this from the point of view of Shannon's information theory. Jaynes wanted a criterion function $H(p_1, p_2, \ldots, p_n)$ that would summarize the uncertainty about the distribution. This is given uniquely by the entropy measure

$$H(p_1, p_2, \ldots, p_n) = -K \sum_{i=1}^{n} p_i \ln(p_i)$$

where $p_i \ln(p_i)$ is defined to be zero if $p_i = 0$ for some positive constant K. The solution to Jaynes's problem is to pick the distribution $(p_1, p_2, \ldots, p_n)$ that maximizes the entropy, subject only to the constraints

$$\begin{aligned} E\{f(x)\} &= \sum_i p_i f(x_i) \\ \sum_i p_i &= 1 \end{aligned}$$

As Golan, Judge, and Miller (1996, 8–10) show, if our knowledge of $E\{f(x)\}$ is based on the outcome of N (very large) trials, then the distribution function $\mathbf{p} = (p_1, p_2, \ldots, p_n)$ that maximizes the entropy measure is the distribution that can give rise to the observed outcomes in the greatest number of ways, which is consistent with what we know. Any other distribution requires more information to justify it. Degenerate distributions, ones where $p_i = 1$ and $p_j = 0$ for $j \neq i$, have entropy of zero. That is to say, they correspond to zero uncertainty and therefore maximal information.

2 Maximum entropy and minimum cross-entropy estimation

More formally, the maximum entropy problem can be represented as

$$\begin{aligned} \max_{\mathbf{p}} H(\mathbf{p}) &= -\sum_{i=1}^{n} p_i \ln(p_i) \\ \text{such that } y_j &= \sum_{i=1}^{n} X_{ji} p_i,\ j = 1, \ldots, J \qquad (1) \\ \sum_{i=1}^{n} p_i &= 1 \qquad (2) \end{aligned}$$

The J constraints given in (1) can be thought of as moment constraints, with y_j being the population mean of the X_j random variable. To solve this problem, we set up the Lagrangian function

$$L = -\mathbf{p}' \ln(\mathbf{p}) - \lambda(\mathbf{X}'\mathbf{p} - \mathbf{y}) - \mu(\mathbf{p}'\mathbf{1} - 1)$$

where $\mathbf{X}$ is the $n \times J$ data matrix,[1] λ is a vector of Lagrange multipliers, and $\mathbf{1}$ is a column vector of ones.

The first-order conditions for an interior solution—that is, one in which the vector $\mathbf{p}$ is strictly positive—are given by

$$\begin{aligned}\frac{\partial L}{\partial \mathbf{p}} &= -\ln(\widehat{\mathbf{p}}) - \mathbf{1} - \mathbf{X}\widehat{\lambda} - \widehat{\mu}\mathbf{1} = \mathbf{0} && (3)\\ \frac{\partial L}{\partial \lambda} &= \mathbf{y} - \mathbf{X}'\widehat{\mathbf{p}} = \mathbf{0} && (4)\\ \frac{\partial L}{\partial \mu} &= 1 - \widehat{\mathbf{p}}'\mathbf{1} = 0 && (5)\end{aligned}$$

These equations can be solved for $\widehat{\lambda}$, and the solution for $\widehat{\mathbf{p}}$ is given by

$$\widehat{\mathbf{p}} = \exp\left(-\mathbf{X}\widehat{\lambda}\right) / \Omega\left(\widehat{\lambda}\right)$$

where

$$\Omega\left(\widehat{\lambda}\right) = \sum_{i=1}^{n} \exp\left(-\mathbf{x}_i\widehat{\lambda}\right)$$

and $\mathbf{x}_i$ is the ith row vector of the matrix $\mathbf{X}$.

The maximum entropy framework can be extended to incorporate prior information about $\mathbf{p}$. Assuming that we have the prior probability distribution $\mathbf{q} = (q_1, q_2, \ldots, q_n)$, then the cross-entropy is defined as (Golan, Judge, and Miller 1996, 11)

$$\begin{aligned}\mathbf{I}(\mathbf{p}, \mathbf{q}) &= \sum_{i=1}^{n} p_i \ln\left(\frac{p_i}{q_i}\right)\\ &= \mathbf{p}'\ln(\mathbf{p}) - \mathbf{p}'\ln(\mathbf{q})\end{aligned}$$

The cross-entropy can be thought of as a measure of the additional information required to go from the distribution $\mathbf{q}$ to the distribution $\mathbf{p}$. The principle of minimum cross-entropy asserts that we should pick the distribution $\mathbf{p}$ that meets the moment constraints (1) and the normalization restriction (2) while requiring the least additional information; that is, we should pick the one that is in some sense closest to $\mathbf{q}$. Formally, we minimize $\mathbf{I}(\mathbf{p}, \mathbf{q})$, subject to the restrictions. Maximum entropy estimation is merely a variant of minimum cross-entropy estimation where the prior $\mathbf{q}$ is the uniform distribution $(1/n, 1/n, \ldots, 1/n)$.

1. In the Golan, Judge, and Miller (1996) book, the constraint is written as $\mathbf{y} = \mathbf{X}\mathbf{p}$, where $\mathbf{X}$ is $J \times n$. For the applications considered below, it is more natural to write the data matrix in the form shown here.

The solution of this problem is given by (Golan, Judge, and Miller 1996, 29)

$$\widetilde{p}_i = q_i \exp\left(\mathbf{x}_i\widetilde{\lambda}\right) / \Omega\left(\widetilde{\lambda}\right) \tag{6}$$

where

$$\Omega\left(\widetilde{\lambda}\right) = \sum_{i=1}^{n} q_i \exp\left(\mathbf{x}_i\widetilde{\lambda}\right) \tag{7}$$

The most efficient way to calculate the estimates is, in fact, not by numerical solution of the first-order conditions [along the lines of (3), (4), and (5)] but by the unconstrained maximization of the dual problem as discussed further in section 3.5.

3 The maxentropy command

3.1 Syntax

The syntax of the `maxentropy` command is

`maxentropy` [*constraint*] *varlist* [*if*] [*in*] `, generate(`*varname* [`, replace`] `)`
    [`prior(`*varname*`) log total(`*#*`) matrix(`*matrix*`)`]

The `maxentropy` command must identify the set of population constraints contained in the **y** vector. These population constraints can be specified either as [*constraint*] or as *matrix* in the `matrix()` option. If neither of these optional arguments is specified, it is assumed that *varlist* is **y** and then **X**.

The command requires that a *varname* be specified in the `generate()` option, in which the estimated **p** vector will be returned.

3.2 Description

`maxentropy` provides minimum cross-entropy or maximum entropy estimates of ill-posed inverse problems, such as the Jaynes's dice problem. The command can also be used to calibrate survey datasets to external totals along the lines of the multiplicative method implemented in the SAS CALMAR macro (Deville and Särndal 1992; Deville, Särndal, and Sautory 1993). This is a generalization of iterative raking as implemented, for instance, in Nick Winter's `survwgt` command, which is available from the Statistical Software Components archive (type `net search survwgt`).

3.3 Options

`generate(`*varname* [`, replace`] `)` provides the variable name in which Stata will store the probability estimates. This must be a new variable name, unless the `replace` suboption is specified, in which case the existing variable is overwritten. `generate()` is required.

`prior(`*varname*`)` requests minimum cross-entropy estimation with the vector of prior probabilities **q** given in the variable *varname*. If `prior()` is not specified, then maximum entropy estimates are returned.

`log` is necessary only if the command is failing to converge. This option specifies to display the output from the maximum likelihood subroutine that is used to calculate the vector λ. The iteration log might provide some diagnostics on what is going wrong.

`total(`#`)` is required if "raising weights" rather than probabilities are desired. The number must be the population total to which the weights are supposed to be summed.

`matrix(`*matrix*`)` passes the constraint vector contained in *matrix*. This must be a column vector that must have as many elements as are given in *varlist*. The order of the constraints in the vector must correspond to the order of the variables given in *varlist*. If no matrix is specified, then `maxentropy` will look for the constraints in the first variable after the command. This variable must have the constraints listed in the first J positions corresponding to the J variables listed in *varlist*.

3.4 Output

`maxentropy` returns output in three forms. First, it returns estimates of the λ coefficients. The absolute magnitude of the coefficient is an indication of how informative the corresponding constraint is, that is, how far it moves the resulting **p** distribution away from the prior **q** distribution in the cross-entropy case or away from the uniform distribution in the maximum entropy case.

Second, the estimates of **p** are returned in the variable specified by the user. Third, the vector of constraints **y** is returned in the matrix `e(constraint)`, with the rows of the matrix labeled according to the variable whose constraint that row represents.

Example

Consider the Jaynes's die problem described earlier. Specifically, let us calculate the probabilities if we know that the mean of the die is 4. We set the problem up by creating the x variable, which contains the discrete distribution of outcomes, that is, $(1, 2, 3, 4, 5, 6)$. The **y** vector contains the mean 4.

```
. set obs 6
obs was 0, now 6
. generate x = _n
. matrix y = (4)
```

(*Continued on next page*)

```
. maxentropy x, matrix(y) generate(p4)
Cross entropy estimates
───────────────────────────
    Variable │    lambda
─────────────┼─────────────
           x │  .17462893
───────────────────────────
p values returned in p4
constraints given in matrix y
```

The λ value corresponding to the constraint $E(x) = 4$ is 0.1746289, so the constraint is informative, that is, the resulting distribution is no longer the uniform one. The message at the end reminds us where the rest of the output is to be obtained (that is, in the `p4` variable) and that the constraints were passed by means of a Stata matrix. To see the **p** estimate itself, we can just `list` the variable:

```
. list x p4, noobs sep(10)

  ┌───────────────┐
  │ x          p4 │
  ├───────────────┤
  │ 1    .1030653 │
  │ 2    .1227305 │
  │ 3     .146148 │
  │ 4    .1740337 │
  │ 5    .2072401 │
  │ 6    .2467824 │
  └───────────────┘
```

The distribution is weighted toward the larger numbers. We can check that these estimates obey the restrictions:

```
. generate xp4=x*p4
. quietly summarize xp4
. display r(sum)
4
```

Finally, we can retrieve a copy of the constraint matrix labeled with the corresponding variables.

```
. matrix list e(constraint)
symmetric e(constraint)[1,1]
   c1
x   4
```

3.5 Methods and formulas

Instead of solving the constrained optimization problem given by the first-order conditions [(3) to (5)] or their cross-entropy analogues, Golan, Judge, and Miller (1996, 30) show that the solution can be found by maximizing the unconstrained dual cross-entropy objective function

$$L(\lambda) = \sum_{j=1}^{J} \lambda_j y_j - \ln\{\mathbf{\Omega}(\lambda)\} = \mathbf{M}(\lambda) \tag{8}$$

where $\mathbf{\Omega}(\lambda)$ is given by (7). Golan, Judge, and Miller show that this function behaves like a maximum likelihood. In this case,

$$\nabla_\lambda \mathbf{M}(\lambda) = \mathbf{y} - \mathbf{X}'\mathbf{p} \tag{9}$$

so that the constraint is met at the point where the gradient is zero. Furthermore,

$$-\frac{\partial^2 \mathbf{M}}{\partial \lambda_j^2} = \sum_{i=1}^{n} p_i x_{ji}^2 - \left(\sum_{i=1}^{n} p_i x_{ji}\right)^2 = \text{var}(x_j) \tag{10}$$

$$-\frac{\partial^2 \mathbf{M}}{\partial \lambda_j \partial \lambda_k} = \sum_{i=1}^{n} p_i x_{ji} x_{ki} - \left(\sum_{i=1}^{n} p_i x_{ji}\right)\left(\sum_{i=1}^{n} p_i x_{ki}\right) = \text{cov}(x_j, x_k) \tag{11}$$

where the variances and covariances are taken with respect to the distribution $\mathbf{p}$. The negative of the Hessian of $\mathbf{M}$ is therefore guaranteed to be positive definite, which guarantees a unique solution provided that the constraints are not inconsistent.

Golan, Judge, and Miller (1996, 25) note that the function $\mathbf{M}$ can be thought of as an expected log likelihood, given the exponential family $\mathbf{p}(\lambda)$ parameterized by λ. Along these lines, we use Stata's maximum likelihood routines to estimate λ, giving it the dual objective function [(8)], gradient [(9)], and negative Hessian [(10) and (11)]. The routine that calculates these is contained in `maxentlambda_d2.ado`. Because of the globally concave nature of the objective function, convergence should be relatively quick, provided that there is a feasible solution in the interior of the parameter space. The command checks for some obvious errors; for example, the population means (y_j) must be inside the range of the X_j variables. If any mean is on the boundary of the range, then a degenerate solution is feasible, but the corresponding Lagrange multiplier will be $\pm\infty$, so the algorithm will not converge.

Once the estimates of λ have been obtained, estimates of $\mathbf{p}$ are derived from (6).

3.6 Saved results

`maxentropy` saves the following in `e()`:

Macros			
`e(cmd)`	`maxentropy`	`e(properties)`	`b V`
Matrices			
`e(b)`	λ coefficient estimates	`e(V)`	inverse of negative Hessian
`e(constraint)`	constraint vector		
Functions			
`e(sample)`	marks estimation sample		

3.7 A cautionary note

The estimation routine treats λ as though it were estimated by maximum likelihood. This is true only if we can write $\mathbf{p}$ as

$$\mathbf{p} \propto \exp\left(-\mathbf{X}\lambda\right)$$

Given that assumption, we could test hypotheses on the λ parameters. Because the estimation routine calculates the inverse of the negative of the Hessian (that is, the asymptotic covariance matrix of λ under this parametric assumption), it would be possible to implement such tests. For most practical applications, this parametric interpretation of the procedure is likely to be dubious.

4 Sample applications

4.1 Jaynes's die problem

In section 3.4, I showed how to calculate the probability distribution given that $\mathbf{y} = 4$. The following code generates predictions given different values for $\mathbf{y}$:

```
matrix y=(2)
maxentropy x, matrix(y) generate(p2)
matrix y=(3)
maxentropy x, matrix(y) generate(p3)
matrix y=(3.5)
maxentropy x, matrix(y) generate(p35)
matrix y=(5)
maxentropy x, matrix(y) generate(p5)
list p2 p3 p35 p4 p5, sep(10)
```

The impact of different prior information on the estimated probabilities is shown in the following table:

```
. list p2 p3 p35 p4 p5, sep(10)

     +-------------------------------------------------------+
     |       p2         p3        p35         p4          p5 |
     |-------------------------------------------------------|
  1. | .4781198   .2467824   .1666667   .1030653    .0205324 |
  2. |  .254752   .2072401   .1666667   .1227305    .0385354 |
  3. |  .135737   .1740337   .1666667    .146148    .0723234 |
  4. | .0723234    .146148   .1666667   .1740337     .135737 |
  5. | .0385354   .1227305   .1666667   .2072401    .2547519 |
  6. | .0205324   .1030652   .1666667   .2467824    .4781198 |
     +-------------------------------------------------------+
```

Note in particular that when we set $\mathbf{y} = 3.5$, the command returns the uniform discrete distribution with $p_i = 1/6$.

We can see the impact of adding in a second constraint by considering the same problem given the population moments

$$\mathbf{y} = \begin{pmatrix} \mu \\ \sigma^2 \end{pmatrix} = \begin{pmatrix} 3.5 \\ \sigma^2 \end{pmatrix}$$

for different values of σ^2. By definition in this case, $\sigma^2 = \sum_{i=1}^{6} p_i \left(x_i - 3.5\right)^2$. We can therefore create the values $\left(x_i - 3.5\right)^2$ and consider which probability distribution $\mathbf{p} = (p_1, p_2, \ldots, p_6)$ will generate both a mean of 3.5 and a given value of σ^2. The code to run this is

```
generate dev2=(x-3.5)^2
matrix y=(3.5 \ (2.5^2/3+1.5^2/3+0.5^2/3))
maxentropy x dev2, matrix(y) generate(pv)
matrix y=(3.5 \ 1)
maxentropy x dev2, matrix(y) generate(pv1)
matrix y=(3.5 \ 2)
maxentropy x dev2, matrix(y) generate(pv2)
matrix y=(3.5 \ 3)
maxentropy x dev2, matrix(y) generate(pv3)
matrix y=(3.5 \ 4)
maxentropy x dev2, matrix(y) generate(pv4)
matrix y=(3.5 \ 5)
maxentropy x dev2, matrix(y) generate(pv5)
matrix y=(3.5 \ 6)
maxentropy x dev2, matrix(y) generate(pv6)
```

with the following final result:

```
. list pv1 pv2 pv pv3 pv4 pv5 pv6, sep(10) noobs

  +----------------------------------------------------------------------------+
  |      pv1        pv2         pv        pv3        pv4        pv5        pv6 |
  |----------------------------------------------------------------------------|
  |  .018632   .0885296   .1666667   .1741325   .2672036   .3659436   .4713601 |
  | .1316041   .1719114   .1666667   .1651027   .1358892   .0896692   .0234196 |
  | .3497639   .2395591   .1666667   .1607649   .0969072   .0443872   .0052203 |
  | .3497639   .2395591   .1666667   .1607649   .0969072   .0443872   .0052203 |
  | .1316041   .1719113   .1666667   .1651026   .1358892   .0896692   .0234196 |
  |  .018632   .0885296   .1666667   .1741325   .2672036   .3659436   .4713601 |
  +----------------------------------------------------------------------------+
```

The probabilities behave as we would expect: in the case where $\sigma^2 = 35/12$, we get the uniform distribution. With variances smaller than this, the probability distribution puts more emphasis on the values 3 and 4, while with higher variances the distribution becomes bimodal with greater probability being attached to extreme values. This output does not reveal that in all cases the λ_1 estimate is basically zero. The reason for this is that with a symmetrical distribution of x_i values around the population mean, the mean is no longer informative and all the information about the distribution of $\mathbf{p}$ derives from the second constraint. If we force $p_4 = p_5 = 0$ so that the distribution is no longer symmetrical, the first constraint becomes informative, as shown in this output:

(*Continued on next page*)

```
. maxentropy x dev2 if x!=5&x!=4, matrix(y) generate(p5, replace)

Cross entropy estimates

────────────────────────────
    Variable │     lambda
─────────────┼──────────────
           x │    .0119916
        dev2 │   .59568007
────────────────────────────

p values returned in p5
constraints given in matrix y

. list x p5 if e(sample), noobs

  ┌──────────────┐
  │ x         p5 │
  ├──────────────┤
  │ 1   .4578909 │
  │ 2   .0427728 │
  │ 3   .0131515 │
  │ 6   .4861848 │
  └──────────────┘
```

This example shows how to overwrite an existing variable and demonstrates that the command allows `if` and `in` qualifiers. It also shows how to use the `e(sample)` function.

4.2 Calibrating a survey

The basic point of calibration is to adjust the sampling weights so that the marginal totals in different categories correspond to the population totals. Typically, the adjustments are made on demographic (for example, age and gender) and spatial variables. Early approaches included iterative raking procedures (Deming and Stephan 1940). These were generalized in the CALMAR routines described in Deville and Särndal (1992). The idea of using a minimum information loss criterion for this purpose is not original (see, for instance, Merz and Stolze [2008]), although it does not seem to have been appreciated that the procedure leads to *identical* estimates as iterative raking-ratio adjustments, if those adjustments are iterated to convergence.

The major advantage of using the cross-entropy approach rather than raking is that it becomes straightforward to incorporate constraints that do not include marginal totals. In many household surveys, for instance, it is plausible that mismatches between the sample and the population arise due to differential success in sampling household types rather than in enumerating individuals within households. Under these conditions, it makes sense to require that all raising weights within a household be identical. I give an example below that shows how cross-entropy estimation with such a constraint can be feasibly implemented.

These capacities also exist within other calibration macros and commands. The advantage of the `maxentropy` command is that it can do so within Stata—and it is fairly easy and quick to use.

To demonstrate these possibilities, we load `example1.dta`, which contains a hypothetical survey with a set of prior weights. The sum of these weights by stratum and gender is given in table 1, where we have also indicated the population totals to which the weights should gross.

Table 1. Sum of weights from `example1.dta` by stratum, gender, and gross weight for population totals

	gender			
stratum	0	1	Margin	Required
0	100	400	500	1600
1	300	200	500	400
Margin	400	600	1000	
Required	1200	800		2000

The weights can be adjusted to these totals by using the downloadable `survwgt` command. To use the `maxentropy` command, we need to convert the desired constraints from population totals into population means. That is straightforward because

$$N = \sum_{i=1}^{n} w_i \tag{12}$$

$$N_{\text{gender}=0} = \sum_{i=1}^{n} w_i \, \mathbf{1}\,(\text{gender} = 0) \tag{13}$$

where $\mathbf{1}\,(\text{gender} = 0)$ is the indicator function. So dividing everything by N, the population total, we get a set of constraints that look identical to those used earlier:

$$\begin{aligned} 1 &= \sum_{i=1}^{n} \frac{w_i}{N} = \sum_{i=1}^{n} p_i \\ \Pr(\text{gender} = 0) &= \sum_{i=1}^{n} \frac{w_i}{N} \, \mathbf{1}\,(\text{gender} = 0) \\ &= \sum_{i=1}^{n} p_i \, \mathbf{1}\,(\text{gender} = 0) \end{aligned}$$

We could obviously add a condition for the proportion where gender = 1, but because of the adding-up constraint, that would be redundant. If we have k categories for a particular variable, we can only use $k - 1$ constraints in our estimation.

In this particular example, the constraint vector is contained in the `constraint` variable. The syntax of the command in this case is

```
maxentropy constraint stratum gender, generate(wt3) prior(weight) total(2000)
```

We did not specify a matrix, so the first variable is interpreted as the constraint vector. We did specify a prior weight and asked Stata to convert the calculated probabilities to raising weights by multiplying them by 2,000. A comparison with the "raked weights" confirms them to be identical in this case.

We can check whether the constraints were correctly rendered by retrieving the constraint matrix used in the estimation:

```
. matrix C=e(constraint)
. matrix list C
C[2,1]
               c1
stratum        .2
 gender  .40000001
```

We see that $E(\texttt{stratum}) = 0.2$ and $E(\texttt{gender}) = 0.4$. Means of dummy variables are, of course, just population proportions; that is, the proportion in `stratum` $= 1$ is 0.2 and the proportion where `gender` $= 1$ is 0.4.

4.3 Imposing constant weights within households

In most household surveys, the household is the unit that is sampled and the individuals are enumerated within it. Consequently, the probability of including an individual conditional on the household being selected is 1. This suggests that the weight attached to every individual within a household should be equal. We can impose this restriction with a fairly simple ploy. We rewrite constraint (12) by first summing over individuals within the household (hhsize) and then summing over households as

$$\begin{aligned} N &= \sum_h \sum_i w_{ih} \\ &= \sum_h \text{hhsize}_h w_h \end{aligned}$$

that is,

$$N = \sum_h w_h^*$$

where w_{ih} is the weight of individual i within household h, equal to the common weight w_h. This constraint can again be written in the form of probabilities as

$$1 = \sum_h \frac{w_h^*}{N}$$

that is,

$$1 = \sum_h p_h^*$$

Consider now any other constraint involving individual aggregates [for example, (13)]

$$\begin{aligned}
N_x &= \sum_{i=1}^{n} w_i x_i \\
&= \sum_h \sum_i w_{ih} x_{ih} \\
&= \sum_h w_h \left(\sum_i x_{ih} \right) \\
\frac{N_x}{N} &= \sum_h \frac{w_h \text{hhsize}_h}{N} \frac{\sum_i x_{ih}}{\text{hhsize}_h}
\end{aligned}$$

Consequently,

$$E(x) = \sum_h p_h^* m_{xh} \tag{14}$$

The term m_{xh} is just the mean of the x variable within household h.

If the prior weight q_h is similarly constant within households (as it should be if it is a design weight), then we similarly create a new variable

$$q_h^* = \text{hhsize}_h \, q_h$$

We can then write the cross-entropy objective function as

$$\begin{aligned}
\mathbf{I}(\mathbf{p}, \mathbf{q}) &= \sum_{i=1}^{n} p_i \ln\left(\frac{p_i}{q_i}\right) = \sum_h \sum_i p_{ih} \ln\left(\frac{p_{ih}}{q_{ih}}\right) \\
&= \sum_h \sum_i p_h \ln\left(\frac{p_h \text{hhsize}_h}{q_h \text{hhsize}_h}\right) \\
&= \sum_h \sum_i p_h \ln\left(\frac{p_h^*}{q_h^*}\right) \\
&= \sum_h \text{hhsize}_h p_h \ln\left(\frac{p_h^*}{q_h^*}\right) \\
&= \sum_h p_h^* \ln\left(\frac{p_h^*}{q_h^*}\right)
\end{aligned}$$

In short, the objective function evaluated over all individuals and imposing the constraint $p_{ih} = p_h$ for all i is identical to the objective function evaluated over households where the probabilities have been adjusted to p_h^* and q_h^*. We therefore run the `maxentropy` command on a household-level file, with the population constraints given by (14). Our cross-entropy estimates can then be retrieved as

$$\widetilde{p}_h = \frac{\widetilde{p}_h^*}{\text{hhsize}_h}$$

We can check that the weights obtained in this way do, in fact, obey all the restrictions—they are obviously constant within household, and when added up over the individuals, they reproduce the required totals.

4.4 Calibrating the South African National Income Dynamics Survey

To assess the performance of the `maxentropy` command on a more realistic problem, we consider the problem of calibrating South Africa's National Income Dynamics Survey. This was a nationally representative sample of around 7,300 households and around 30,000 individuals. From the sampling design, a set of "design weights" were calculated, but application of these weights to the realized sample led to a severe undercount when compared with the official population estimates.

The calibration was to be done to reproduce the nine provincial population counts and 136 age $\times$ sex $\times$ race cell totals. One practical difficulty that was immediately encountered was how to treat individuals where age, sex, or race information was missing, because this category does not exist in the national estimates. It was decided to keep the relative weights of the missing observations constant through the calibration, creating a 137th age, sex, and race category. From each group of dummy variables, one category had to be omitted, creating altogether 144 (or $8 + 136$) constraints.

`hhcollapsed.dta` contains household-level means of all these variables plus the household design weights. The code to create cross-entropy weights that are constant within households is given by the following:

```
use hhcollapsed
maxentropy constraint P1-WFa80, prior(q) generate(hw) total(48687000)
replace hw=hw/hhsize
matrix list e(constraint)
```

With 144 constraints and 7,305 observations, the command took 18 seconds to calculate the new weights on a standard desktop computer.

In this context, the λ estimates prove informative. The output of the command is

```
. maxentropy constraint P1-WFa80, prior(q) generate(hw) total(48687000)

Cross entropy estimates

─────────────────────────
     Variable │    lambda
──────────────┼──────────
           P1 │ -.15945276
           P2 │  .00735986
           P3 │  .14000206
  (output omitted)
        IMa75 │  15.402056
        IMa80 │  8.6501559
        IFa_0 │ -7.0753612
        IFa_5 │  2.3584972
  (output omitted)
        IFa75 │ -9.2778495
        IFa80 │  14.142518
  (output omitted)
        WFa70 │  .05009103
        WFa75 │  .90961156
        WFa80 │  4.6868009
─────────────────────────

p values returned in hw
constraints given in variable constraints
```

The huge coefficients for old Indian males and old Indian females suggests that the population constraints affected the weights for these categories substantially. Given the large number of constraints, mistakes are possible. The easiest way to check that the command has worked correctly is to add up the weights within categories and to check that they add up to the intended totals. Listing the constraint matrix used by the command is also a useful check. In this case, the labeling of the rows does help:

```
. matrix list e(constraint)

e(constraint)[144,1]
              c1
   P1  .10803039
   P2  .13514069
   P3  .02320805
   P4  .05914972
   P5  .20764017
   P6  .07044568
   P7  .21462312
   P8  .07373177
AMa_0  .04486157
AMa_5  .04584822
  (output omitted)
WFa75   .0012318
WFa80  .00147087
```

The first eight constraints are the province proportions followed by the proportions in the age, sex, and race cells.

5 Conclusion

This article introduced the power of maximum entropy and minimum cross-entropy estimation. The `maxentropy` command uses Stata's powerful maximum-likelihood estimation routines to provide fast estimates of even complicated problems. I have shown how the command can be used to calibrate a survey to a set of known population totals while imposing restrictions like constant weights within households.

6 References

Deming, W. E., and F. F. Stephan. 1940. On a least squares adjustment of a sample frequency table when the expected marginal totals are known. *Annals of Mathematical Statistics* 11: 427–444.

Deville, J.-C., and C.-E. Särndal. 1992. Calibration estimators in survey sampling. *Journal of the American Statistical Association* 87: 376–382.

Deville, J.-C., C.-E. Särndal, and O. Sautory. 1993. Generalized raking procedures in survey sampling. *Journal of the American Statistical Association* 88: 1013–1020.

Golan, A., G. G. Judge, and D. Miller. 1996. *Maximum Entropy Econometrics: Robust Estimation with Limited Data.* Chichester, UK: Wiley.

Jaynes, E. T. 1957. Information theory and statistical mechanics. *Physical Review* 106: 620–630.

Merz, J., and H. Stolze. 2008. Representative time use data and new harmonised calibration of the American Heritage Time Use Data 1965−1999. *electronic International Journal of Time Use Research* 5: 90–126.

Mittelhammer, R. C., G. G. Judge, and D. J. Miller. 2000. *Econometric Foundations.* Cambridge: Cambridge University Press.

About the author

Martin Wittenberg teaches core econometrics and microeconometrics to graduate students in the Economics Department at the University of Cape Town.