b
UNIVERSITÄT
BERN

Faculty of Business, Economics and Social Sciences

Department of Social Sciences

University of Bern Social Sciences Working Paper No. 35

Influence functions continued. A framework for estimating standard errors in reweighting, matching, and regression adjustment

Ben Jann

March 28, 2020

http://ideas.repec.org/p/bss/wpaper/35.html
http://econpapers.repec.org/paper/bsswpaper/35.htm

University of Bern
Department of Social Sciences
Fabrikstrasse 8
CH-3012 Bern

Tel. +41 (0)31 631 48 11
Fax +41 (0)31 631 48 17
info@sowi.unibe.ch
www.sowi.unibe.ch

Influence functions continued. A framework for estimating standard errors in reweighting, matching, and regression adjustment

Ben Jann

March 28, 2020

Abstract

In Jann (2019) I provided some reflections on influence functions for linear regression (with an application to regression adjustment). Based on an analogy to variance estimation in the generalized method of moments (GMM), I extend the discussion in this paper to maximum-likelihood models such as logistic regression and then provide influence functions for a variety of treatment effect estimators such as inverse-probability weighting (IPW), regression adjustment (RA), inverse-probability weighted regression adjustment (IPWRA), exact matching (EM), Mahalanobis distance matching (MD), and entropy balancing (EB). The goal of this exercise is to provide a framework for standard error estimation in all these estimators.

Keywords: Influence function, sampling variance, standard error, generalized method of moments, maximum likelihood, logistic regression, inverse-probability weighting, inverse-probability weighted regression adjustment, exact matching, Mahalanobis distance matching, entropy balancing, average treatment effect, causal inference

JEL classification: C01, C12, C13, C21, C25, C31, C83, C87

Contents

1 Introduction

Influence functions are awesome because they can be used for standard error estimation. Loosely speaking, an influence function quantifies how a statistic changes, once a small amount of data mass is added at a certain point in the probability distribution on which the statistic is based (see Hampel, 1974, who established the concept).[1] From a more applied perspective, an influence function provides an approximation of the "effect" of an observation on a statistic computed from the data.

It has been shown that, asymptotically, the sampling variance of a statistic is equal to the sampling variance of the mean of its influence function (see, e.g., Hampel et al., 1986, p. 85, Staudte and Sheather, 1990, p. 79–81, Deville, 1999). That is, once you know the influence function, you get the standard errors for free. Furthermore, if you have a series of different statistics, whatever they may be, and you know the influence function for each of them, you can simply estimate the joint variance matrix across all statistics by computing the variance matrix of the influence functions. This is extremely useful because it allows you to freely combine estimates from different models and subpopulations and conduct whatever tests you like. Finally, influence function are fully compatible with Taylor-linearized variance estimation for complex surveys.

The challenge, of course, is to derive the influence functions. In the statistical literature there are many examples, but (1) you have to find them and (2) you have to understand them (the latter I typically find harder then the former). Furthermore, it might be that you are interested in an estimator for which no one ever worked out the influence function. It may thus be good to know how to do this.

In this paper I first show how the generalized method of moments (GMM) provides a framework that makes it relatively easy to derive influence functions, also for people who are not trained statisticians (like me). I also show the close connection between GMM and maximum-likelihood estimation and illustrate how influence functions can be obtained for arbitrary maximum-likelihood models, even without knowing any math. In the second

[1] Influence functions are extensively used in robust statistics, see, e.g., Hampel et al. (1986).

part I then discuss influence functions for a variety of treatment effect estimators under the conditional independence assumption, such as inverse-probability weighting, regression adjustment, matching, and entropy balancing. My ultimate goal is to develop standard error estimation methods to be implemented in Stata command `kmatch` (Jann, 2017).

2 Using GMM to derive influence functions

2.1 General approach

It appears that the generalized method of moments (GMM) – or, in fact, just the method of moments (MM), as I will only consider cases that are exactly identified (that is, models in which the number of moment condition equals the number of parameters) – can be helpful to derive influence functions. Drawing on the exposition in the Stata manual (see [R] **gmm**; for a textbook exposition see, e.g., Cameron and Trivedi, 2005), GMM can be described as a procedure to find estimate $\hat{\theta}$ that solves the following system of moment equations

$$E(\mathbf{X}_i'\mathbf{u}_i(\theta)) = \mathbf{0}$$

with

$$\mathbf{X}_i = \begin{bmatrix} x_{i1} & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & x_{i2} & \dots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \dots & x_{iq} \end{bmatrix} \quad \text{and} \quad \mathbf{u}_i(\theta) = \begin{bmatrix} u_{i1}(\theta_1) \\ u_{i2}(\theta_2) \\ \vdots \\ u_{iq}(\theta_q) \end{bmatrix} \quad \text{and} \quad \theta = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_q \end{bmatrix}$$

where x_{ij} is a $1 \times k_j$ vector of predictors, θ_j is a $k_j \times 1$ vector of parameters, and $u_{ij}(\theta_j)$ is a (scalar) residual or error term associated with moment equation j.[2] The total number of parameters in the system is $k = \sum_{j=1}^{q} k_j$, which is also the total number of predictors in an exactly identified model (typically including a constant in each equation). An alternative,

[2] I do not make a distinction between instruments z_{ij} and predictors x_{ij} here because in the cases I will consider the two are the same.

slightly more flexible representation of GMM uses $\mathbf{h}_i(\mathbf{X}_i;\theta)$ in place of $\mathbf{X}_i'\mathbf{u}_i(\theta)$ such that the optimization problem can be written as

$$E(\mathbf{h}_i(\mathbf{X}_i;\theta)) = \mathbf{0} \qquad \text{with} \qquad \mathbf{h}_i(\mathbf{X}_i;\theta) = \begin{bmatrix} h_{i1}(x_{i1};\theta_1) \\ h_{i2}(x_{i2};\theta_2) \\ \vdots \\ h_{iq}(x_{iq};\theta_q) \end{bmatrix}$$

where each moment equation $h_{ij}(x_{ij};\theta_j) = x_i' u_{ij}(\theta_j)$ is a $k_j \times 1$ vector of moment conditions. I will use this alternative representation in this paper as I find it somewhat easier to handle and because not all estimators discussed below can be written in the error-term representation.

In an exactly identified GMM model, where GMM's weighting matrix has no impact, the (robust) $k \times k$ variance matrix of $\hat{\theta}$ is estimated as

$$\widehat{V}(\hat{\theta}) = \frac{1}{N}\left(\overline{\mathbf{G}}(\hat{\theta})^{-1}\right)\left(\frac{1}{N}\sum_{i=1}^{N}\mathbf{h}_i(\mathbf{X}_i;\hat{\theta})\mathbf{h}_i(\mathbf{X}_i;\hat{\theta})'\right)\left(\overline{\mathbf{G}}(\hat{\theta})^{-1}\right)'$$

where the inner part of the equation is the moment covariance matrix and

$$\overline{\mathbf{G}}(\hat{\theta}) = \frac{1}{N}\sum_{i=1}^{N}\left.\frac{\partial \mathbf{h}_i(\mathbf{X}_i;\theta)}{\partial \theta'}\right|_{\theta=\hat{\theta}}$$

is the Jacobian matrix of the moment equations.[3] Now, for the same model, let $\lambda_i(\hat{\theta})$ be a $k \times 1$ vector of observation i's values of the influence function for $\hat{\theta}$. According to the literature on influence functions,

$$\hat{V}(\hat{\theta}) = \frac{1}{N}\left(\frac{1}{N}\sum_{i=1}^{N}\lambda_i(\hat{\theta})\lambda_i(\hat{\theta})'\right)$$

is a consistent estimate of the variance matrix of $\hat{\theta}$. Rearranging $\widehat{V}(\hat{\theta})$ from GMM to

$$\hat{V}(\hat{\theta}) = \frac{1}{N}\left(\frac{1}{N}\sum_{i=1}^{N}\left(\overline{\mathbf{G}}(\hat{\theta})^{-1}\mathbf{h}_i(\mathbf{X}_i;\hat{\theta})\right)\left(\overline{\mathbf{G}}(\hat{\theta})^{-1}\mathbf{h}_i(\mathbf{X}_i;\hat{\theta})\right)'\right)$$

suggests that we can derive the influence function as

$$\lambda_i(\hat{\theta}) = \overline{\mathbf{G}}(\hat{\theta})^{-1}\mathbf{h}_i(\mathbf{X}_i;\hat{\theta}) \tag{1}$$

[3] I am puzzled by what $\partial \mathbf{h}_i(\mathbf{X}_i;\theta)/\partial\theta'|_{\theta=\hat{\theta}}$ means, but it seems to boil down to $-\partial \mathbf{h}_i(\mathbf{X}_i;\hat{\theta})/\partial\hat{\theta}'$.

Defined in this way, the variance matrix computed from the influence function will be identical to the variance matrix computed by GMM.

▷ **Example**

As a simple example and proof of concept, consider a standard (single-equation) linear regression model

$$y_i = x_i\beta + \epsilon_i$$

In this case, the residual is $u_i(\beta) = y_i - x_i\beta$ such that $h_i(x_i;\beta) = x_i'(y_i - x_i\beta)$. Since $\partial h_i(x_i;\beta)/\partial\beta' = -x_i'x_i$, we have

$$\overline{\mathbf{G}}(\hat{\beta}) = \frac{1}{N}\sum_{i=1}^{N} x_i'x_i$$

and, hence,

$$\lambda_i(\hat{\beta}) = \left(\frac{1}{N}\sum_{i=1}^{N} x_i'x_i\right)^{-1} x_i'(y_i - x_i\hat{\beta}) \tag{2}$$

This is equivalent to the expression for the influence function of linear regression given in Jann (2019; also see Kahn, 2015; jayk, 2015).

◁

2.2 Constructing influence functions recursively

In models with multiple moment equations, the computation of $\overline{\mathbf{G}}(\hat{\theta})$ is not particularly convenient. However, we can piece things together equation by equation. Let

$$\overline{\mathbf{G}}_{lj} = \frac{1}{N}\sum_{i=1}^{N} \left.\frac{\partial h_{il}(x_{il};\theta_l)}{\partial\theta_j'}\right|_{\theta=\hat{\theta}}$$

which is typically easy to compute without looping over observations using cross products. We can then assemble $\overline{\mathbf{G}}(\hat{\theta})$ as

$$\overline{\mathbf{G}}(\hat{\theta}) = \begin{bmatrix} \overline{\mathbf{G}}_{11} & \overline{\mathbf{G}}_{12} & \dots & \overline{\mathbf{G}}_{1q} \\ \overline{\mathbf{G}}_{21} & \overline{\mathbf{G}}_{22} & \dots & \overline{\mathbf{G}}_{2q} \\ \vdots & \vdots & \ddots & \vdots \\ \overline{\mathbf{G}}_{q1} & \overline{\mathbf{G}}_{q2} & \dots & \overline{\mathbf{G}}_{qq} \end{bmatrix}$$

Note that $\overline{\mathbf{G}}_{lj}$ will be $\mathbf{0}$ if equation l is first-order independent from equation j, that is, if parameters from equation j do not directly appear in equation l (this will be true for all elements above the diagonal if the moment equations are arranged in an order such that earlier equations do not depend on later equations, which is always possible in a recursive system). This also means that influence functions for selected elements of θ can be constructed recursively without having to evaluate the complete inverse of $\overline{\mathbf{G}}(\hat{\theta})$. For example, in a two-equation system

$$\mathbf{h}_i(\mathbf{X}_i;\theta) = \begin{bmatrix} h_{i1}(x_{i1};\theta_1) \\ h_{i2}(x_{i2};\theta_2) \end{bmatrix} \qquad \text{such that} \qquad \overline{\mathbf{G}}(\hat{\theta}) = \begin{bmatrix} \overline{\mathbf{G}}_{11} & \mathbf{0} \\ \overline{\mathbf{G}}_{21} & \overline{\mathbf{G}}_{22} \end{bmatrix}$$

we can obtain the influence functions for $\hat{\theta}_1$ and $\hat{\theta}_2$ sequentially as

$$\lambda_i(\hat{\theta}_1) = \overline{\mathbf{G}}_{11}^{-1} h_{i1}(x_{i1};\hat{\theta}_1) \tag{3}$$

and

$$\lambda_i(\hat{\theta}_2) = \overline{\mathbf{G}}_{22}^{-1} \Big(h_{i2}(x_{i2};\hat{\theta}_2) + (-\overline{\mathbf{G}}_{21})\lambda_i(\hat{\theta}_1) \Big) \tag{4}$$

This shows that the influence function for $\hat{\theta}_2$ has two components: a first component that is equal to the influence function we would get if θ_1 was assumed fixed, plus a correction term accounting for the fact that θ_1 is estimated. The correction term is simply a transformation of the influence function for $\hat{\theta}_1$, where the transformation is determined by the (average) "effect" of $\hat{\theta}_1$ on $\hat{\theta}_2$ (i.e. the expectation of the derivative of $h_{i2}(x_{i2};\hat{\theta}_2)$ by $\hat{\theta}_1$). Likewise, in a three-equation system

$$\mathbf{h}_i(\mathbf{X}_i;\theta) = \begin{bmatrix} h_{i1}(x_{i1};\theta_1) \\ h_{i2}(x_{i2};\theta_2) \\ h_{i3}(x_{i3};\theta_3) \end{bmatrix} \qquad \text{such that} \qquad \overline{\mathbf{G}}(\hat{\theta}) = \begin{bmatrix} \overline{\mathbf{G}}_{11} & \mathbf{0} & \mathbf{0} \\ \overline{\mathbf{G}}_{21} & \overline{\mathbf{G}}_{22} & \mathbf{0} \\ \mathbf{0} & \overline{\mathbf{G}}_{32} & \overline{\mathbf{G}}_{33} \end{bmatrix}$$

the influence function for $\hat{\theta}_3$ can be written as

$$\lambda_i(\hat{\theta}_3) = \overline{\mathbf{G}}_{33}^{-1} \Big(h_{i3}(x_{i3};\hat{\theta}_3) + (-\overline{\mathbf{G}}_{32})\lambda_i(\hat{\theta}_2) \Big) \tag{5}$$

with $\lambda_i(\hat{\theta}_2)$ as in Equation 4 (and $\lambda_i(\hat{\theta}_1)$ as in Equation 3). This nicely illustrates how the correction terms propagate throughout the recursive system.

2.3 Application to logistic regression

Consider a logistic regression of $d_i \in \{0, 1\}$ on predictors x_i (including a constant). The model is defined as

$$\Pr(d_i = 1|x_i) = p_i = \frac{\exp(x_i\beta)}{1 + \exp(x_i\beta)}$$

Parameters β can be estimated using GMM by setting the moment equation to

$$h_i(x_i; \beta) = x_i'(d_i - p_i) = x_i' \left(d_i - \frac{\exp(x_i\beta)}{1 + \exp(x_i\beta)} \right)$$

Since

$$\frac{\partial h_i(x_i; \hat{\beta})}{\partial \hat{\beta}'} = -x_i'\hat{p}_i(1 - \hat{p}_i)x_i$$

the influence function is given as

$$\lambda_i(\hat{\beta}) = \left(\frac{1}{N} \sum_{i=1}^{N} x_i'\hat{p}_i(1 - \hat{p}_i)x_i \right)^{-1} x_i'(d_i - \hat{p}_i) \qquad \text{with } \hat{p}_i = \frac{\exp(x_i\hat{\beta})}{(1 + \exp(x_i\hat{\beta})} \tag{6}$$

This is easy to compute, as is illustrated in the following example (using some rearrangement such that the influence function can be computed for all observations in a single line):

```
. sysuse auto, clear
(1978 Automobile Data)
. logit foreign price weight
  (output omitted)
. mata:
------------------------------------------------- mata (type end to exit) -----
: N    = st_nobs()
: D    = st_data(.,"foreign")
: X    = st_data(.,"price weight"), J(N, 1, 1)
: b    = st_matrix("e(b)")'
: p    = invlogit(X * b)
: h    = X :* (D :- p)
: Ginv = invsym(cross(X, (p :* (1 :- p)), X) / N)
: IF   = h * Ginv'
: b, mean(IF)', sqrt(diagonal(variance(IF) / N))
                  1              2              3
  +-------------------------------------------------+
1 |    .0009295971    3.05631e-13    .0002562202    |
2 |   -.0058785402   -1.81294e-12    .0016624897    |
  +-------------------------------------------------+
```

```
  3 |    9.000473365     2.47677e-09     2.852253138
```

```
: end
```

By design, the mean of the influence function across all observations is zero, which is true in the example apart from roundoff error (see the second column in the matrix displayed in the above output). For the standard errors of the logit coefficients, we get values of $\mathrm{SE}(\hat{\beta}_1) = .0002562$, $\mathrm{SE}(\hat{\beta}_2) = .0016625$, and $\mathrm{SE}(\hat{\beta}_3) = 2.852253$. These are exactly the values returned by `logit` with the `robust` option:

```
. logit foreign price weight, nolog robust
Logistic regression                             Number of obs     =         74
                                                Wald chi2(2)      =      13.29
                                                Prob > chi2       =     0.0013
Log pseudolikelihood = -17.976341               Pseudo R2         =     0.6008

------------------------------------------------------------------------------
             |               Robust
     foreign |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
       price |   .0009296   .0002562     3.63   0.000     .0004274    .0014318
      weight |  -.0058785   .0016625    -3.54   0.000     -.009137   -.0026201
       _cons |   9.000473   2.852253     3.16   0.002      3.41016    14.59079
------------------------------------------------------------------------------
```

We could also employ `gmm` to estimate the model. Note that the moment equations have to be specified using the error-term notation in `gmm`, that is, we have to split $h_i(x_i;\beta) = x_i'(d_i - p_i)$ into a (scalar) residual term $u_i(\beta) = (d_i - p_i)$ and vector of instruments x_i (to which `gmm` will automatically add a constant). In our example, the command then looks as follows:

```
. gmm (foreign - invlogit({xb:price weight} + {b0})), ///
>     instruments(price weight) nolog

Final GMM criterion Q(b) =  2.50e-30

note: model is exactly identified

GMM estimation

Number of parameters =   3
Number of moments    =   3
Initial weight matrix: Unadjusted                 Number of obs   =         74
GMM weight matrix:     Robust

------------------------------------------------------------------------------
             |               Robust
             |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
```

```
------------------------------------------------------------------------------
       price |   .0009296   .0002545     3.65   0.000     .0004308    .0014284
      weight |  -.0058785   .0016512    -3.56   0.000    -.0091149   -.0026422
-------------+----------------------------------------------------------------
         /b0 |   9.000473   2.832916     3.18   0.001     3.448059    14.55289
------------------------------------------------------------------------------
Instruments for equation 1: price weight _cons
```

The standard errors reported by `gmm` are slightly different from the standard errors based on the influence function. This is just a scaling issue: `gmm` divides by N instead of $N-1$ when computing the moment covariance matrix. Therefore, standard errors by `gmm` differ by a factor of $\sqrt{N/(N-1)}$. To obtain the same result as `gmm` simply multiply the influence function or the resulting standard errors by $\sqrt{(N-1)/N}$:

```
. mata:
------------------------------------------------- mata (type end to exit) -----
: sqrt(diagonal(variance(IF * sqrt((N-1)/N)) / N)),
> sqrt(diagonal(variance(IF) / N)) * sqrt((N-1)/N)
                 1             2
  +-------------------------------+
1 |  .0002544831   .0002544831    |
2 |  .0016512185   .0016512185    |
3 |  2.832915606   2.832915606    |
  +-------------------------------+

: end
------------------------------------------------------------------------------
```

2.4 Application to (other) maximum-likelihood models

Logistic regression is typically estimated by the maximum likelihood (ML) method, but there is a close connection between GMM and ML. While GMM solves the moment conditions

$$\frac{1}{N}\sum_{i=1}^{N}\mathbf{h}_i(\mathbf{X}_i;\theta)=\mathbf{0}$$

ML solves

$$\frac{1}{N}\sum_{i=1}^{N}\frac{\partial \ell_i(\theta)}{\partial\theta}=\mathbf{0}$$

where $\ell_i(\theta)$ is observation i's contribution to the log likelihood. For logistic regression $\partial\ell_i(\theta)/\partial\theta$ is equal to $x_i'(d_i-p_i)$, such that ML and GMM lead to the same result if the GMM moment equation is defined as $\mathbf{h}_i(\mathbf{X}_i;\theta)=x_i'(d_i-p_i)$ with $p_i=\exp(x_i\theta)/(1+\exp(x_i\theta))$.

Depending on the definition of $\mathbf{h}_i(\mathbf{X}_i;\theta)$, however, GMM and ML may also differ. A prominent example is the probit model. If the moment equation, as would be intuitive, is defined as $\mathbf{h}_i(\mathbf{X}_i;\theta) = x_i'(d_i - p_i) = x_i'(d_i - \Phi(x_i\theta))$, where $\Phi()$ is the standard normal distribution function, then GMM yields slightly different results than `probit`. This is because the first derivative of the log likelihood of the probit model is *not* equal to $x_i'(d_i - \Phi(x_i\theta))$. A more complicated moment equation has to be used for GMM to produce the same result as `probit`, namely

$$\mathbf{h}_i(\mathbf{X}_i;\theta) = x_i' \left(d_i \frac{\phi(x_i\theta)}{\Phi(x_i\theta)} - (1-d_i)\frac{\phi(x_i\theta)}{1-\Phi(x_i\theta)} \right)$$

where $\phi()$ is the standard normal density function (see Drukker, 2014).

From a general perspective, GMM will be consistent with ML if $\mathbf{h}_i(\mathbf{X}_i;\theta)$ corresponds to the gradient vector of the maximum-likelihood model (the vector of first derivatives of the log likelihood). In this case, $\overline{\mathbf{G}}(\theta)$ corresponds to the information matrix, the negative of the expectation of the Hessian (the matrix of second derivatives of the log likelihood). That is,

$$\hat{\lambda}(\hat{\theta}) = \left(\sum_{i=1}^{N} -\frac{\partial \ell_i(\hat{\theta})}{\partial\hat{\theta}\partial\hat{\theta}'} \right)^{-1} \frac{\partial \ell_i(\hat{\theta})}{\partial\hat{\theta}} \tag{7}$$

where $\ell_i(\theta)$ is again observation i's contribution to the log likelihood.

In maximum likelihood estimation, the inverse of the observed information matrix (divided by N) is typically used as the default (non-robust) estimate of the variance matrix (method `vce(oim)` in Stata; see [R] ***vce_option***). Hence, instead of working out the details for each model, we can simply use the default (non-robust, model-based) variance matrix returned by the estimation command (multiplied by N), together with equation-level scores computed by `predict` (multiplied by x_i to obtain parameter-levels scores), to compute the influence function. For logistic regression, this would go as follows:

```
. sysuse auto, clear
(1978 Automobile Data)
. logit foreign price weight
  (output omitted)
. predict sc, score
```

```
. mata:
------------------------------------------------------- mata (type end to exit) -----
: N    = st_nobs()
: X    = st_data(.,"price weight"), J(N, 1, 1)
: b    = st_matrix("e(b)")'
: h    = X :* st_data(.,"sc")
: Ginv = st_matrix("e(V)") * N
: IF   = h * Ginv'
: b, mean(IF)', sqrt(diagonal(variance(IF) / N))
                  1             2             3
  +-------------------------------------------------------------+
1 |    .0009295971   -1.63677e-12    .0002562202                |
2 |   -.0058785402   -3.26293e-11    .0016624897                |
3 |    9.000473365    8.98807e-08    2.852253109                |
  +-------------------------------------------------------------+
: end
--------------------------------------------------------------------------------
```

Results are the same as above (apart from roundoff error). The same procedure can be used for other models, such as `probit`:

```
. sysuse auto, clear
(1978 Automobile Data)
. probit foreign price weight, nolog
  (output omitted)
. predict sc, score
. mata:
------------------------------------------------------- mata (type end to exit) -----
: N    = st_nobs()
: X    = st_data(.,"price weight"), J(N, 1, 1)
: b    = st_matrix("e(b)")'
: h    = X :* st_data(.,"sc")
: Ginv = st_matrix("e(V)") * N
: IF   = h * Ginv'
: b, mean(IF)', sqrt(diagonal(variance(IF) / N))
                  1             2             3
  +-------------------------------------------------------------+
1 |    .0005169548    2.60476e-12    .0001247663                |
2 |   -.0032380468   -1.56645e-11    .0007976984                |
3 |    4.921935132    2.54878e-08    1.411308829                |
  +-------------------------------------------------------------+
: end
--------------------------------------------------------------------------------
```

```
. probit foreign price weight, nolog robust

Probit regression                                 Number of obs     =         74
                                                  Wald chi2(2)      =      17.71
                                                  Prob > chi2       =     0.0001
Log pseudolikelihood = -18.006571                 Pseudo R2         =     0.6001

------------------------------------------------------------------------------
             |               Robust
     foreign |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
       price |    .000517   .0001248     4.14   0.000     .0002724    .0007615
      weight |   -.003238   .0007977    -4.06   0.000    -.0048015   -.0016746
       _cons |   4.921935   1.411309     3.49   0.000     2.155821     7.68805
------------------------------------------------------------------------------
```

We see that the influence function successfully reproduces the robust standard errors reported by `probit`. The procedure also works for models with ancillary parameters or multiple-equations, although the code gets slightly more complicated because there are multiple score variables. Here is an example for the ordered probit model:

```
. webuse fullauto, clear
(Automobile Models)

. oprobit rep77 foreign length mpg

  (output omitted)

. keep if e(sample)
(8 observations deleted)

. predict sc*, score

. mata:
------------------------------------------------- mata (type end to exit) -----
: N    = st_nobs()

: X    = st_data(.,"foreign length mpg")

: b    = st_matrix("e(b)")'

: u    = st_data(.,"sc*")

: h    = X :* u[,1], u[|1,2 \ .,.|]

: Ginv = st_matrix("e(V)") * N

: IF   = h * Ginv'

: b, mean(IF)', sqrt(diagonal(variance(IF) / N))
                 1              2              3
    +-------------------------------------------+
  1 |   1.704860532   -3.84566e-07    .4504827633 |
  2 |   .0468675272   -1.65855e-08    .0128720281 |
  3 |   .1304559141   -4.31039e-08    .0432849128 |
  4 |   10.15890352   -4.09244e-06    3.256835182 |
    +-------------------------------------------+
```

```
  5 |   11.21002954   -4.11106e-06    3.229300565
  6 |   12.54560994   -4.15534e-06    3.272299618
  7 |   13.98059347   -4.21373e-06    3.387610162
```

```
: end
```

```
. oprobit rep77 foreign length mpg, nolog robust

Ordered probit regression                       Number of obs     =         66
                                                Wald chi2(3)      =      26.86
                                                Prob > chi2       =     0.0000
Log pseudolikelihood = -78.020025               Pseudo R2         =     0.1321
```

rep77	Coef.	Robust Std. Err.	z	P>|z|	[95% Conf.	Interval]
foreign	1.704861	.4504827	3.78	0.000	.8219306	2.58779
length	.0468675	.012872	3.64	0.000	.0216388	.0720962
mpg	.1304559	.0432849	3.01	0.003	.0456191	.2152928
/cut1	10.1589	3.256835			3.775625	16.54218
/cut2	11.21003	3.2293			4.880718	17.53934
/cut3	12.54561	3.272299			6.132022	18.9592
/cut4	13.98059	3.38761			7.341001	20.62019

Likewise, here is an example for the multinomial logit model:

```
. webuse sysdsn1, clear
(Health insurance data)

. mlogit insure age male nonwhite i.site

  (output omitted)

. keep if e(sample)
(29 observations deleted)

. predict sc*, score

. mata:
```

```
: N    = st_nobs()

: X    = st_data(.,"age male nonwhite i.site"), J(N, 1, 1)

: b    = st_matrix("e(b)")'

: u    = st_data(.,"sc*")

: h    = X :* u[,1], X :* u[,2], X :* u[,3]

: Ginv = st_matrix("e(V)") * N

: IF   = h * Ginv'

: select((b, mean(IF)', sqrt(diagonal(variance(IF) / N))), b:!=0)
                  1             2             3
```

1	-.0117449879	-1.08926e-10	.0061442367
2	.5616934289	1.00903e-08	.2036455151
3	.9747767878	-9.27211e-09	.2382791848
4	.1130358552	-1.10819e-08	.2120257922
5	-.5879879357	-1.12641e-08	.2312904752
6	.2697126985	1.53193e-08	.3268667528
7	-.0077961365	-7.14357e-12	.0109023022
8	.4518496351	1.58174e-08	.3614143078
9	.217058946	-2.42229e-09	.4246704968
10	-1.211562719	-1.81485e-08	.4807049716
11	-.2078123246	-1.20144e-08	.3611922444
12	-1.286942952	2.09729e-09	.599982016

```
: end
```

```
. mlogit insure age male nonwhite i.site, nolog robust

Multinomial logistic regression                 Number of obs     =        615
                                                Wald chi2(10)     =      37.40
                                                Prob > chi2       =     0.0000
Log pseudolikelihood = -534.36165               Pseudo R2         =     0.0387
```

insure	Coef.	Robust Std. Err.	z	P>|z|	[95% Conf.	Interval]
Indemnity	(base outcome)					
Prepaid						
age	-.011745	.0061442	-1.91	0.056	-.0237875	.0002975
male	.5616934	.2036455	2.76	0.006	.1625556	.9608313
nonwhite	.9747768	.2382792	4.09	0.000	.5077582	1.441795
site						
2	.1130359	.2120258	0.53	0.594	-.3025271	.5285988
3	-.5879879	.2312905	-2.54	0.011	-1.041309	-.1346669
_cons	.2697127	.3268668	0.83	0.409	-.3709344	.9103598
Uninsure						
age	-.0077961	.0109023	-0.72	0.475	-.0291643	.013572
male	.4518496	.3614143	1.25	0.211	-.2565094	1.160209
nonwhite	.2170589	.4246705	0.51	0.609	-.6152799	1.049398
site						
2	-1.211563	.480705	-2.52	0.012	-2.153727	-.2693983
3	-.2078123	.3611922	-0.58	0.565	-.9157361	.5001115
_cons	-1.286943	.599982	-2.14	0.032	-2.462886	-.1109998

Again, standard errors obtained from the influence functions are identical to the robust standard errors returned by the maximum-likelihood command.

The fact that influence functions can be obtained in this way for maximum-likelihood models may be good to know. In fact, Stata's `svy` prefix command (see [SVY] **svy**) does something very similar. It is, however, not of much use to me here because I am interested in other models that can not be handled in this way.

3 Influence functions for treatment effect estimators

3.1 Estimands

Let d_i be a treatment indicator, where $d_i = 1$ denotes that some treatment was received or some intervention has been administered, and $d_i = 0$ denotes the absence of treatment. In causal inference, we are interested in the effect that treatment D has on outcome Y. If we just compare the observed outcomes of those who received treatment and those who did not, we may be misguided, because treatment assignment could have been selective (unless treatment has been randomized). Conceptually, we are therefore interested in potential outcomes y_i^1 (the outcome that would be observed under treatment) and y_i^0 (the outcome that would be observed without treatment), and define the treatment effect for unit i as the difference between the two, $\delta_i = y_i^1 - y_i^0$. We cannot, however, observe both potential outcomes at the same time. In particular, y_i^0 remains unobserved for those who received treatment, and y_i^1 remains unobserved for those who did not receive treatment. We thus need a credible strategy to "impute" these values. Furthermore, we typically do not focus on treatment effects for individual units, but some aggregate such as their average in a given population.[4]

[4]For a textbook exposition on the potential outcomes framework an the estimation of treatment effects, see Morgan and Winship (2015). Other prominent references are, for example, Imbens and Rubin (2015); Angrist and Pischke (2009); Imbens and Wooldridge (2009).

In particular, three different treatment effect estimands are often of interest. The (unconditional) average treatment effect (ATE), average treatment effect on the treated (ATT), and the average treatment effect on the untreated (ATC). Let $p = \Pr(D = 1)$ be the (unconditional) treatment probability and let

$$\eta_1^1 = E(Y^1|D = 1) \qquad \eta_0^1 = E(Y^1|D = 0) \qquad \eta^1 = E(Y^1) = p\,\eta_1^1 + (1-p)\eta_0^1$$
$$\eta_1^0 = E(Y^0|D = 1) \qquad \eta_0^0 = E(Y^0|D = 0) \qquad \eta^0 = E(Y^0) = p\,\eta_1^0 + (1-p)\eta_0^0$$

Then the ATT (δ^1) and the ATC (δ^0) are

$$\delta^1 = E(Y^1 - Y^0|D = 1) = E(Y^1|D = 1) - E(Y^0|D = 1) = \eta_1^1 - \eta_1^0$$
$$\delta^0 = E(Y^1 - Y^0|D = 0) = E(Y^1|D = 0) - E(Y^0|D = 0) = \eta_0^1 - \eta_0^0$$

Furthermore, the ATT (δ) is

$$\begin{aligned}\delta &= E(Y^1 - Y^0) = E(Y^1) - E(Y^0) = \eta^1 - \eta^0 \\ &= p\,\eta_1^1 + (1-p)\eta_0^1 - (p\,\eta_1^0 + (1-p)\eta_0^0) \\ &= p(\eta_1^1 - \eta_1^0) + (1-p)(\eta_0^1 - \eta_0^0) \\ &= p\,\delta^1 + (1-p)\delta^0\end{aligned}$$

Estimation of p (the unconditional treatment probability), η_1^1 (the observed outcome mean among treated), and η_0^0 (the observed outcome mean among untreated) is typically unproblematic, as all necessary information is observed. The challenge in causal inference is to find good estimates for the counterfactual potential outcome means η_1^0 (the mean outcome among treated had they not been treated) and η_0^1 (the mean outcome among untreated had they been treated).

A variety of estimators for this problem have been proposed in the literature under the so called conditional independence assumption

$$(Y^1, Y^0) \perp\!\!\!\perp D \,|\, X$$

where X is a vector of covariates. Prominent examples are matching, inverse-probability weighting, or regression adjustment. Essentially, these estimators assume that, within the

strata defined by the unique values of X, treatment D is random. Therefore, these estimators can be seen as different variants of an attempt to stratify on X when estimating η_1^0 and η_0^1. Some of the estimators do so indirectly by controlling for X in a treatment assignment model (modeling effects of X on D), others control for X directly in the outcome model (modeling effects of X on Y), and some of the estimators do both. Below I will derive influence functions for a variety of these estimators.

3.2 Inverse-probability weighting

Average treatment effect on the treated (ATT)

First consider the inverse-probability weighting (IPW) estimator of the average treatment effect on the treated (ATT). Three quantities are of interest: The mean outcome in the treatment group, $\hat{\eta}_1^1$, the (counterfactual) potential outcome mean without treatment in the treatment group, $\hat{\eta}_1^0$, and the difference between these two quantities, $\hat{\delta}^1$ (the estimate of the ATT). The IPW approach estimates these quantities as follows:

$$\hat{\eta}_1^1 = \frac{1}{\sum_i d_i} \sum_i d_i y_i \qquad \hat{\eta}_1^0 = \frac{1}{\sum_i \hat{\omega}_i^0} \sum_i \hat{\omega}_i^0 y_i \qquad \hat{\delta}^1 = \hat{\eta}_1^1 - \hat{\eta}_1^0$$

where d_i is a treatment indicator (1 for the treated, 0 for controls) and $\hat{\omega}_i^0$ is as described below. Since $\hat{\eta}_1^1$ is a simple mean, its influence function is trivial:

$$\lambda_i(\hat{\eta}_1^1) = \frac{N}{\sum_i d_i}(y_i - \hat{\eta}_1^1) \tag{8}$$

Furthermore, the influence function for the ATT is given as

$$\lambda_i(\hat{\delta}^1) = \lambda_i(\hat{\eta}_1^1) - \lambda_i(\hat{\eta}_1^0) \tag{9}$$

The derivation of $\lambda_i(\hat{\eta}_1^0)$ is more challenging. The potential outcome mean $\hat{\eta}_1^0$ is estimated by taking a reweighted average of the outcome values in the control group. The weights are derived in a way such that the reweighted control group looks as similar as possible to

the treatment group in terms of the distribution of covariates. More specifically, define the weights as

$$\hat{\omega}_i^0 = (1 - d_i)\frac{\hat{p}_i}{1 - \hat{p}_i} = \begin{cases} 0 & \text{if } d_i = 1 \\ \hat{p}_i/(1 - \hat{p}_i) & \text{if } d_i = 0 \end{cases}$$

with the propensity score $\hat{p}_i$ estimated using logistic regression such that

$$\hat{p}_i = \frac{\exp(x_i\hat{\beta})}{1 + \exp(x_i\hat{\beta})}$$

where x_i is $1 \times k$ vector of predictors (including a constant).[5] Formulating the estimator for $\theta = (\beta', \eta_1^0)'$ as a GMM problem yields the moment equations

$$\mathbf{h}_i(\mathbf{X}_i; \theta)) = \begin{bmatrix} h_{i1}(x_i; \beta) \\ h_{i2}(1; \eta_1^0) \end{bmatrix} = \begin{bmatrix} x_i'(d_i - p_i) \\ \omega_i^0(y_i - \eta_1^0) \end{bmatrix}$$

where $p_i = \exp(x_i\beta)/(1 + \exp(x_i\beta))$ and $\omega_i^0 = (1 - d_i)\ p_i/(1 - p_i)$. Since

$$\frac{\partial p_i}{\partial \beta'} = p_i(1 - p_i)x_i \qquad \frac{\partial \omega_i^0}{\partial \beta'} = (1 - d_i)\frac{p_i}{1 - p_i}x_i = \omega_i^0 x_i$$

we get

$$\overline{\mathbf{G}}(\hat{\theta}) = \frac{1}{N}\sum_{i=1}^{N} \begin{bmatrix} x_i'\hat{p}_i(1 - \hat{p}_i)x_i & \mathbf{0} \\ -\hat{\omega}_i^0(y_i - \hat{\eta}_1^0)x_i & \hat{\omega}_i^0 \end{bmatrix}$$

Hence, based on the recursive expression in Equation 4, we can write the influence function for $\hat{\eta}_1^0$ as

$$\lambda_i(\hat{\eta}_1^0) = \frac{N}{\sum_i \hat{\omega}_i^0}\left(\hat{\omega}_i^0(y_i - \hat{\eta}_1^0) - \overline{\mathbf{G}}_{21}\overline{\mathbf{G}}_{11}^{-1}x_i'(d_i - \hat{p}_i)\right) \tag{10}$$

where

$$\overline{\mathbf{G}}_{21} = \frac{1}{N}\sum_{i=1}^{N} -\hat{\omega}_i^0(y_i - \hat{\eta}_1^0)x_i \qquad \text{and} \qquad \overline{\mathbf{G}}_{11} = \frac{1}{N}\sum_{i=1}^{N} x_i'\hat{p}_i(1 - \hat{p}_i)x_i$$

Example:

[5] An alternative would be to use a probit model. In this case, the definition $h_{i1}(x_i; \beta)$ and the definition of p, as well as the derivatives of $h_{i1}(x_i; \beta)$ and $h_{i2}(1; \beta)$ with respect to β have to be replaced. The general structure of the following solution, however, remains the same.

```
. sysuse auto, clear
(1978 Automobile Data)

. logit foreign price weight

  (output omitted)

. mata:
------------------------------------------------- mata (type end to exit) -----
: N        = st_nobs()

: D        = st_data(.,"foreign")

: X        = st_data(.,"price weight"), J(N, 1, 1)

: Y        = st_data(.,"mpg")

: // compute IF of eta01
: b        = st_matrix("e(b)")'

: p        = invlogit(X * b)

: h1       = X :* (D - p)

: G11inv   = invsym(cross(X, p :* (1 :- p), X) / N)

: w0       = p :/ (1 :- p) :* !D

: eta01    = mean(Y, w0)

: h2       = w0 :* (Y :- eta01)

: G21      = colsum(-h2 :* X) / N

: IF_eta01 = N/sum(w0) * (h2 - h1 * G11inv' * G21')

: // compute IF for eta11
: eta11    = mean(Y, D)

: IF_eta11 = N/sum(D) * D :* (Y :- eta11)

: // compute IF for ATT
: ATT      = eta11 - eta01

: IF_ATT   = IF_eta11 - IF_eta01

: // display results (point estimate, mean of IF, standard error)
: (ATT, eta11, eta01)', mean((IF_ATT, IF_eta11, IF_eta01))',
>     sqrt(diagonal(variance((IF_ATT, IF_eta11, IF_eta01)) / N)) * sqrt((N-1)/N)
                 1              2              3

  1 |  -4.855450742   -9.79714e-10    2.114228039
  2 |   24.77272727   -6.54131e-16    1.377102927
  3 |   29.62817801    9.79714e-10     1.77167096

: end
------------------------------------------------------------------------------
```

We can compare the results to `teffects`:

```
. teffects ipw (mpg) (foreign price weight), nolog atet
Treatment-effects estimation                    Number of obs     =         74
```

```
Estimator      : inverse-probability weights
Outcome model  : weighted mean
Treatment model: logit
```

mpg	Coef.	Robust Std. Err.	z	P>|z|	[95% Conf.	Interval]
ATET						
foreign (Foreign vs Domestic)	-4.855451	2.114228	-2.30	0.022	-8.999262	-.7116399
POmean						
foreign Domestic	29.62818	1.771671	16.72	0.000	26.15577	33.10059

The results are identical (note that $\hat{\eta}_1^1$ is not shown in output of `teffects`).

Average treatment effect on the untreated (ATC)

The influence function for the average treatment effect on the untreated (ATC) can be obtained analogously. The following three quantities are of interest:

$$\hat{\eta}_0^1 = \frac{1}{\sum_i \hat{\omega}_i^1} \sum_i \hat{\omega}_i^1 y_i \qquad \hat{\eta}_0^0 = \frac{1}{\sum_i (1-d_i)} \sum_i (1-d_i) y_i \qquad \hat{\delta}^0 = \hat{\eta}_0^1 - \hat{\eta}_0^0$$

where the weights $\hat{\omega}_i^1$ are defined as

$$\hat{\omega}_i^1 = d_i \frac{1-\hat{p}_i}{\hat{p}_i} = \begin{cases} (1-\hat{p}_i)/\hat{p}_i & \text{if } d_i = 1 \\ 0 & \text{if } d_i = 0 \end{cases}$$

To compute the influence function for $\hat{\eta}_0^1$, replace the moment equation $h_{i2}(1;\eta_1^0)$ in the above discussion by

$$h_{i2}(1;\eta_0^1) = \omega_i^1 (y_i - \eta_0^1)$$

such that

$$\overline{\mathbf{G}}(\hat{\theta}) = \frac{1}{N} \sum_{i=1}^{N} \begin{bmatrix} x_i' \hat{p}_i (1-\hat{p}_i) x_i & \mathbf{0} \\ \hat{\omega}_i^1 (y_i - \hat{\eta}_0^1) x_i & \hat{\omega}_i^1 \end{bmatrix}$$

The influence function for $\hat{\eta}_0^1$ then is

$$\lambda_i(\hat{\eta}_0^1) = \frac{N}{\sum_i \hat{\omega}_i^1} \left(\hat{\omega}_i^1 (y_i - \hat{\eta}_0^1) - \overline{\mathbf{G}}_{21} \overline{\mathbf{G}}_{11}^{-1} x_i' (d_i - \hat{p}_i) \right) \tag{11}$$

with

$$\overline{\mathbf{G}}_{21} = \frac{1}{N}\sum_{i=1}^{N} \hat{\omega}_i^1 (y_i - \hat{\eta}_0^1) x_i \qquad \text{and} \qquad \overline{\mathbf{G}}_{11} = \frac{1}{N}\sum_{i=1}^{N} x_i' \hat{p}_i (1 - \hat{p}_i) x_i$$

The computation goes as follows (reusing some results from above):

```
. mata:
------------------------------------------------- mata (type end to exit) -----
: // compute IF of eta10
: w1       = (1 :- p) :/ p :* D

: eta10    = mean(Y, w1)

: h2       = w1 :* (Y :- eta10)

: G21      = colsum(h2 :* X) / N

: IF_eta10 = N/sum(w1) * (h2 - h1 * G11inv' * G21')

: // compute IF for eta00 and ATC
: eta00    = mean(Y, !D)

: IF_eta00 = N/sum(!D) * !D :* (Y :- eta00)

: // compute IF for  ATC
: ATC      = eta10 - eta00

: IF_ATC   = IF_eta10 - IF_eta00

: // display results (point estimate, mean of IF, standard error)
: (ATC, eta10, eta00)', mean((IF_ATC, IF_eta10, IF_eta00))',
>     sqrt(diagonal(variance((IF_ATC, IF_eta10, IF_eta00)) / N)) * sqrt((N-1)/N)
                  1              2              3
  +-------------------------------------------------------+
1 |    2.996205655   -6.63793e-10    2.072139979          |
2 |    22.82312873   -6.63792e-10    2.157750915          |
3 |    19.82692308    2.80556e-16    .6514214963          |
  +-------------------------------------------------------+

: end
------------------------------------------------------------------------------
```

We can confirm the results for the ATC using `teffects` by flipping treatment and control, although note that the ATC will have a wrong sign:

```
. teffects ipw (mpg) (foreign price weight), nolog atet tlevel(0)

Treatment-effects estimation                    Number of obs     =         74
Estimator      : inverse-probability weights
Outcome model  : weighted mean
Treatment model: logit
------------------------------------------------------------------------------
             |               Robust
         mpg |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
ATET         |
     foreign |
```

(Domestic vs Foreign)	-2.996206	2.07214	-1.45	0.148	-7.057525	1.065114
POmean foreign						
Foreign	22.82313	2.157751	10.58	0.000	18.59401	27.05224

Average treatment effect (ATE)

Finally, the average treatment effect (ATE) is defined as

$$ATE = \Pr(D=1)ATT + \Pr(D=0)ATC$$

and can be estimated as

$$\hat{\delta} = \hat{p}\,\hat{\delta}^1 + (1-\hat{p})\hat{\delta}^0 = \hat{p}(\hat{\eta}_1^1 - \hat{\eta}_1^0) + (1-\hat{p})(\hat{\eta}_0^1 - \hat{\eta}_0^0)$$

were $\hat{p} = \frac{1}{N}\sum_{i=1}^{N} d_i$. If the treatment probability $\Pr(D=1)$ is assumed fixed, then the influence function for the ATE would simply be:

$$\lambda_i(\hat{\delta}) = \hat{p}\lambda_i(\hat{\delta}^1) + (1-\hat{p})\lambda_i(\hat{\delta}^0) = \hat{p}(\lambda_i(\hat{\eta}_1^1) - \lambda_i(\hat{\eta}_1^0)) + (1-\hat{p})(\lambda_i(\hat{\eta}_0^1) - \lambda_i(\hat{\eta}_0^0))$$

Typically, however, we would want to account for the additional uncertainty due to the variability of the group sizes in our estimate. The influence function for the treatment probability is

$$\lambda_i(\hat{p}) = d_i - \hat{p}$$

Using the chain rule (see Jann, 2019), we get

$$\begin{aligned}\lambda_i(\hat{\delta}) &= \hat{p}\lambda_i(\hat{\delta}^1) + (1-\hat{p})\lambda_i(\hat{\delta}^0) + (\hat{\delta}^1 - \hat{\delta}^0)(d_i - \hat{p}) \\ &= \hat{p}(\lambda_i(\hat{\eta}_1^1) - \lambda_i(\hat{\eta}_1^0)) + (1-\hat{p})(\lambda_i(\hat{\eta}_0^1) - \lambda_i(\hat{\eta}_0^0)) + ((\hat{\eta}_1^1 - \hat{\eta}_1^0) - (\hat{\eta}_0^1 - \hat{\eta}_0^0))(d_i - \hat{p})\end{aligned} \tag{12}$$

Continuing the above example, the results for the ATE are as follows:

```
. mata:
------------------------------------------------- mata (type end to exit) -----
: p      = sum(D) / N
```

```
: ATE    = p * ATT + (1 - p) * ATC
: IF_ATE = p * IF_ATT :+ (1 - p) * IF_ATC :+ (ATT - ATC) * (D :- p)
: ATE, mean(IF_ATE), sqrt(variance(IF_ATE) / N) * sqrt((N-1)/N)
                  1              2              3

  1 |    .6619294285   -7.57715e-10    1.825943322

: end
```

Once again, we can compare our results to `teffects`:

```
. teffects ipw (mpg) (foreign price weight), nolog

Treatment-effects estimation                    Number of obs     =         74
Estimator      : inverse-probability weights
Outcome model  : weighted mean
Treatment model: logit
```

mpg	Coef.	Robust Std. Err.	z	P>|z|	[95% Conf.	Interval]
ATE						
foreign (Foreign vs Domestic)	.5362646	1.71922	0.31	0.755	-2.833344	3.905873
POmean						
foreign Domestic	23.55664	1.241046	18.98	0.000	21.12423	25.98904

Surprisingly, neither the estimate for the ATE, nor the standard error are exactly the same. This is because `teffects` computes the ATE in a direct way using weights equivalent to $1+\hat{\omega}_i^1+\hat{\omega}_i^0$ instead of applying the above formula. That is, `teffects` computes the ATE as a weighted mean difference using weights $1+(1-\hat{p}_i)/\hat{p}_i = 1/\hat{p}_i$ in the treatment group and weights $1+\hat{p}_i/(1-\hat{p}_i) = 1/(1-\hat{p}_i)$ in the control group. The result would be the same as in the indirect approach above if $\sum_i \hat{\omega}^1$ was equal to the size of the control group and $\sum_i \hat{\omega}^0$

was equal to the size of the treatment group, but this only holds approximately in a finite sample. In terms of GMM, the ATE estimator employed by `teffects` can be written as

$$\mathbf{h}_i(\mathbf{X}_i;\theta)) = \begin{bmatrix} h_{i1}(x_i;\beta) \\ h_{i2}(1;\eta^0) \\ h_{i3}(1;\eta^1) \end{bmatrix} = \begin{bmatrix} x_i'(d_i - p_i) \\ \tilde{\omega}_i^0(y_i - \eta^0) \\ \tilde{\omega}_i^1(y_i - \eta^1) \end{bmatrix}$$

with $\tilde{\omega}_i^0 = (1-d_i)/(1-p_i)$ and $\tilde{\omega}_i^1 = d_i/p_i$ such that the influence functions are

$$\lambda_i(\hat{\eta}^0) = \frac{N}{\sum_i \hat{\tilde{\omega}}_i^0}\left(\hat{\tilde{\omega}}_i^0(y_i - \hat{\eta}^0) - \overline{\mathbf{G}}_{21}\overline{\mathbf{G}}_{11}^{-1}x_i'(d_i - p_i)\right) \tag{13}$$

$$\lambda_i(\hat{\eta}^1) = \frac{N}{\sum_i \hat{\tilde{\omega}}_i^1}\left(\hat{\tilde{\omega}}_i^1(y_i - \hat{\eta}^1) - \overline{\mathbf{G}}_{31}\overline{\mathbf{G}}_{11}^{-1}x_i'(d_i - p_i)\right) \tag{14}$$

with

$$\overline{\mathbf{G}}_{21} = \frac{1}{N}\sum_{i=1}^{N} -\hat{\tilde{\omega}}_i^0(y_i - \hat{\eta}^0)\hat{p}_i x_i \qquad \text{and} \qquad \overline{\mathbf{G}}_{31} = \frac{1}{N}\sum_{i=1}^{N} \hat{\tilde{\omega}}_i^1(y_i - \hat{\eta}^1)(1-\hat{p}_i)x_i$$

and $\overline{\mathbf{G}}_{11}$ as above. The ATE is then defined as $\hat{\delta} = \hat{\eta}^1 - \hat{\eta}^0$ with influence function

$$\lambda_i(\hat{\delta}) = \lambda_i(\hat{\eta}^1) - \lambda_i(\hat{\eta}^0) \tag{15}$$

Using these formulas, we can replicate the results by `teffects` as follows:

```
. sysuse auto, clear
(1978 Automobile Data)

. logit foreign price weight

  (output omitted)

. mata:
------------------------------------------------- mata (type end to exit) -----
: N       = st_nobs()

: D       = st_data(.,"foreign")

: X       = st_data(.,"price weight"), J(N, 1, 1)

: Y       = st_data(.,"mpg")

: // compute influence function for eta0
: b       = st_matrix("e(b)")'

: p       = invlogit(X * b)

: h1      = X :* (D - p)

: G11inv  = invsym(cross(X, p :* (1 :- p), X) / N)
```

```
: w0      = !D :/ (1 :- p)
: eta0    = mean(Y, w0)
: h2      = w0 :* (Y :- eta0)
: G21     = colsum(-h2 :* p :* X) / N
: IF_eta0 = N/sum(w0) * (h2 - h1 * G11inv' * G21')
: // compute influence function for eta1
: w1      = D :/ p
: eta1    = mean(Y, w1)
: h3      = w1 :* (Y :- eta1)
: G31     = colsum(h3 :* (1 :- p) :* X) / N
: IF_eta1 = N/sum(w1) * (h3 - h1 * G11inv' * G31')
: // compute influence function for ATE
: ATE     = eta1 - eta0
: IF_ATE  = IF_eta1 - IF_eta0
: // display results (point estimate, mean of IF, standard error)
: (ATE, eta1, eta0)', mean((IF_ATE, IF_eta1, IF_eta0))',
>     sqrt(diagonal(variance((IF_ATE, IF_eta1, IF_eta0)) / N)) * sqrt((N-1)/N)
                 1              2              3

  1    .5362645719   -1.56794e-09    1.719219768
  2    24.09290314   -3.24039e-10    1.454267747
  3    23.55663857    1.24390e-09    1.241046052

: end
```

3.3 Regression adjustment

Average treatment effect on the treated (ATT)

In regression adjustment, the estimator for the average treatment effect on the treated (ATT) is defined as

$$\hat{\delta}^1 = \hat{\eta}_1^1 - \hat{\eta}_1^0$$

with

$$\hat{\eta}_1^1 = \frac{1}{\sum_i d_i} \sum_i d_i y_i \qquad \text{and} \qquad \hat{\eta}_1^0 = \frac{1}{\sum_i d_i} \sum_i d_i (y_i - z_i \hat{\gamma}_0)$$

where $\hat{\gamma}_0$ is estimated by regressing Y on predictors Z in the control group. The GMM representation for $\theta = (\gamma_0', \eta_1^0)'$ is

$$\mathbf{h}_i(\mathbf{X}_i; \theta) = \begin{bmatrix} h_{i1}(z_i; \gamma_0) \\ h_{i2}(1; \eta_1^0) \end{bmatrix} = \begin{bmatrix} z_i'(1 - d_i)(y_i - z_i\gamma_0) \\ d_i(z_i\gamma_0 - \eta_1^0) \end{bmatrix}$$

such that

$$\overline{\mathbf{G}}(\hat{\theta}) = \frac{1}{N} \sum_{i=1}^{N} \begin{bmatrix} z_i'(1 - d_i)z_i & \mathbf{0} \\ -d_i z_i & d_i \end{bmatrix}$$

Based on Equation 4, this lead to the following influence function:

$$\lambda_i(\hat{\eta}_1^0) = \frac{N}{\sum_i d_i} \left(d_i(z_i\hat{\gamma}_0 - \hat{\eta}_1^0) - \overline{\mathbf{G}}_{21}\overline{\mathbf{G}}_{11}^{-1} z_i'(1 - d_i)(y_i - z_i\hat{\gamma}_0) \right) \tag{16}$$

with

$$\overline{\mathbf{G}}_{21} = \frac{1}{N} \sum_{i=1}^{N} -d_i z_i \qquad \text{and} \qquad \overline{\mathbf{G}}_{11} = \frac{1}{N} \sum_{i=1}^{N} z_i'(1 - d_i)z_i$$

Example:

```
. sysuse auto, clear
(1978 Automobile Data)
. regress mpg price weight if foreign==0
  (output omitted)
. mata:
------------------------------------------------- mata (type end to exit) -----
: N   = st_nobs()
: D   = st_data(.,"foreign")
: Z   = st_data(.,"price weight"), J(N, 1, 1)
: Y   = st_data(.,"mpg")
: // compute IF for eta01
: g0     = st_matrix("e(b)")'
: Zg0    = Z * g0
: h1     = Z :* !D :* (Y - Zg0)
: G11inv = invsym(cross(Z, !D, Z) / N)
: eta01  = mean(Zg0, D)
: h2     = D :* (Zg0 :- eta01)
: G21    = colsum(-D :* Z) / N
: IF_eta01 = N/sum(D) * (h2 - h1 * G11inv' * G21')
: // compute IF for eta11
```

```
: eta11    = mean(Y, D)

: IF_eta11 = N/sum(D) * D :* (Y :- eta11)

: // compute IF for ATT
: ATT = eta11 - eta01

: IF_ATT  = IF_eta11 - IF_eta01

: // display results (point estimate, mean of IF, standard error)
: (ATT, eta11, eta01)', mean((IF_ATT, IF_eta11, IF_eta01))',
>     sqrt(diagonal(variance((IF_ATT, IF_eta11, IF_eta01)) / N)) * sqrt((N-1)/N)
                  1              2              3

  1    -1.820520622    1.52956e-15     1.41055613
  2     24.77272727   -6.54131e-16     1.377102927
  3     26.59324789   -2.25795e-15     1.00111591

: end
```

```
. teffects ra (mpg price weight) (foreign), nolog atet

Treatment-effects estimation                    Number of obs     =         74
Estimator      : regression adjustment
Outcome model  : linear
Treatment model: none
```

mpg	Coef.	Robust Std. Err.	z	P>|z|	[95% Conf.	Interval]
ATET						
foreign (Foreign vs Domestic)	-1.820521	1.410556	-1.29	0.197	-4.58516	.9441185
POmean						
foreign Domestic	26.59325	1.001116	26.56	0.000	24.6311	28.5554

Results are the same as computed by `teffects`.

Average treatment effect on the untreated (ATC)

The regression adjustment estimator for the average treatment effect on the untreated (ATC) is the reverse: estimate a regression in the treatment group and then make out-of-sample predictions in the control group to obtain $\hat{\eta}_0^1$. Therefore,

$$\mathbf{h}_i(\mathbf{X}_i;\theta) = \begin{bmatrix} h_{i1}(z_i;\gamma_1) \\ h_{i2}(1;\eta_0^1) \end{bmatrix} = \begin{bmatrix} z_i' d_i (y_i - z_i\gamma_1) \\ (1-d_i)(z_i\gamma_1 - \eta_0^1) \end{bmatrix}$$

such that

$$\overline{\mathbf{G}}(\hat{\theta}) = \frac{1}{N}\sum_{i=1}^{N} \begin{bmatrix} z_i' d_i z_i & \mathbf{0} \\ (d_i - 1)z_i & 1 - d_i \end{bmatrix}$$

and, hence,

$$\lambda_i(\hat{\eta}_0^1) = \frac{N}{\sum_i (1-d_i)} \Big((1-d_i)(z_i\hat{\gamma}_1 - \hat{\eta}_0^1)) - \overline{\mathbf{G}}_{21}\overline{\mathbf{G}}_{11}^{-1} z_i' d_i (y_i - z_i\hat{\gamma}_1) \Big) \tag{17}$$

with

$$\overline{\mathbf{G}}_{21} = \frac{1}{N}\sum_{i=1}^{N}(d_i - 1)z_i \qquad \text{and} \qquad \overline{\mathbf{G}}_{11} = \frac{1}{N}\sum_{i=1}^{N} z_i' d_i z_i$$

Example:

```
. regress mpg price weight if foreign==1
  (output omitted)
. mata:
------------------------------------------------- mata (type end to exit) -----
: // compute IF for eta01
: g1       = st_matrix("e(b)")'
: Zg1      = Z * g1
: h1       = Z :* D :* (Y - Zg1)
: G11inv   = invsym(cross(Z, D, Z) / N)
: eta10    = mean(Zg1, !D)
: h2       = !D :* (Zg1 :- eta10)
: G21      = colsum(- !D :* Z) / N
: IF_eta10 = N/sum(!D) * (h2 - h1 * G11inv' * G21')
: // compute IF for eta00
: eta00    = mean(Y, !D)
: IF_eta00 = N/sum(!D) * !D :* (Y :- eta00)
: // compute IF for ATC
```

```
: ATC    = eta10 - eta00
: IF_ATC = IF_eta10 - IF_eta00
: // display results (point estimate, mean of IF, standard error)
: (ATC, eta10, eta00)', mean((IF_ATC, IF_eta10, IF_eta00))',
>     sqrt(diagonal(variance((IF_ATC, IF_eta10, IF_eta00)) / N)) * sqrt((N-1)/N)
                  1              2              3
  1 |  -3.726390889   -9.64094e-15    4.555759314
  2 |   16.10053219   -9.34472e-15    4.622990113
  3 |   19.82692308    2.80556e-16    .6514214963

: end
```

```
. teffects ra (mpg price weight) (foreign), nolog atet tlevel(0) // = ATC * -1

Treatment-effects estimation                    Number of obs     =         74
Estimator      : regression adjustment
Outcome model  : linear
Treatment model: none
```

mpg	Coef.	Robust Std. Err.	z	P>|z|	[95% Conf.	Interval]
ATET foreign (Domestic vs Foreign)	3.726391	4.555759	0.82	0.413	-5.202733	12.65552
POmean foreign Foreign	16.10053	4.62299	3.48	0.000	7.039638	25.16143

Average treatment effect (ATE)

Finally, the influence function for the ATE can be computed in the usual way (see page 23):

```
. mata:
                                                 mata (type end to exit)
: p      = sum(D) / N
: ATE    = p * ATT + (1 - p) * ATC
: IF_ATE = p * IF_ATT :+ (1 - p) * IF_ATC :+ (ATT - ATC) * (D :- p)
: ATE, mean(IF_ATE), sqrt(variance(IF_ATE) / N) * sqrt((N-1)/N)
                  1              2              3
  1 |  -3.15978081   -6.75361e-15    3.281466003
```

```
: end
```

```
. teffects ra (mpg price weight) (foreign), nolog
Treatment-effects estimation                    Number of obs     =         74
Estimator      : regression adjustment
Outcome model  : linear
Treatment model: none
```

mpg	Coef.	Robust Std. Err.	z	P>|z|	[95% Conf.	Interval]
ATE						
foreign (Foreign vs Domestic)	-3.159781	3.281466	-0.96	0.336	-9.591336	3.271774
POmean						
foreign Domestic	21.83853	.727837	30.00	0.000	20.412	23.26507

3.4 Inverse-probability-weighted regression adjustment

Average treatment effect on the treated (ATT)

In inverse-probability-weighted regression adjustment, the estimator for the average treatment effect on the treated (ATT) is defined as

$$\hat{\delta}^1 = \hat{\eta}_1^1 - \hat{\eta}_1^0$$

with

$$\hat{\eta}_1^1 = \frac{1}{\sum_i d_i} \sum_i d_i y_i \qquad \text{and} \qquad \hat{\eta}_1^0 = \frac{1}{\sum_i d_i} \sum_i d_i (y_i - z_i \hat{\gamma}_0)$$

where $\hat{\gamma}_0$ is estimated by regressing Y on Z in the control group while applying weights $\hat{\omega}_i^0$ that have been estimated as explained above for inverse probability weighting (again, probit could be used as an alternative, which would require that we replace some of the expressions below), that is

$$\hat{\omega}_i^0 = (1 - d_i) \frac{\hat{p}_i}{1 - \hat{p}_i} \qquad \text{with} \qquad \hat{p}_i = \frac{\exp(x_i \hat{\beta})}{1 + \exp(x_i \hat{\beta})}$$

The GMM problem for $\theta = (\beta', \gamma_0', \eta_1^0)'$ then is

$$\mathbf{h}_i(\mathbf{X}_i; \theta) = \begin{bmatrix} h_{i1}(x_i; \beta) \\ h_{i2}(z_i; \gamma_0) \\ h_{i3}(1; \eta_1^0) \end{bmatrix} = \begin{bmatrix} x_i'(d_i - p_i) \\ z_i'\omega_i^0(y_i - z_i\gamma_0) \\ d_i(z_i\gamma_0 - \eta_1^0) \end{bmatrix}$$

such that

$$\overline{\mathbf{G}}(\hat{\theta}) = \frac{1}{N}\sum_{i=1}^{N} \begin{bmatrix} x_i'\hat{p}_i(1-\hat{p}_i)x_i & \mathbf{0} & \mathbf{0} \\ -z_i'\hat{\omega}_i^0(y_i - z_i\hat{\gamma}_0)x_i & z_i'\hat{\omega}_i^0 z_i & \mathbf{0} \\ \mathbf{0} & -d_i z_i & d_i \end{bmatrix}$$

Using the recursive formula in Equation 5, this leads to the following influence function:

$$\lambda_i(\hat{\eta}_1^0) = \frac{N}{\sum_i d_i}\left(d_i(z_i\hat{\gamma}_0 - \hat{\eta}_1^0) - \overline{\mathbf{G}}_{32}\overline{\mathbf{G}}_{22}^{-1}\left(z_i'\hat{\omega}_i^0(y_i - z_i\hat{\gamma}_0) - \overline{\mathbf{G}}_{21}\overline{\mathbf{G}}_{11}^{-1}x_i'(d_i - \hat{p}_i)\right)\right) \quad (18)$$

with

$$\overline{\mathbf{G}}_{32} = \frac{1}{N}\sum_{i=1}^{N} -d_i z_i \qquad \overline{\mathbf{G}}_{21} = \frac{1}{N}\sum_{i=1}^{N} -z_i'\hat{\omega}_i^0(y_i - z_i\hat{\gamma}_0)x_i$$

$$\overline{\mathbf{G}}_{22} = \frac{1}{N}\sum_{i=1}^{N} z_i'\hat{\omega}_i^0 z_i \qquad \overline{\mathbf{G}}_{11} = \frac{1}{N}\sum_{i=1}^{N} x_i'\hat{p}_i(1-\hat{p}_i)x_i$$

The computation goes as follows:

```
. sysuse auto, clear
(1978 Automobile Data)

. mata:
------------------------------------------------- mata (type end to exit) -----
: Dnm = "foreign"; Xnm = "price weight"

: Ynm = "mpg"    ; Znm = "price weight turn"

: N   = st_nobs()

: D   = st_data(., Dnm); X = st_data(., Xnm), J(N, 1, 1)

: Y   = st_data(., Ynm); Z = st_data(., Znm), J(N, 1, 1)

: // estimate logit and create weights
: stata("quietly logit " + Dnm + " " + Xnm)

: p        = invlogit(X * st_matrix("e(b)")')

: w0       = p :/ (1 :- p) :* !D

: st_store(., st_addvar("double", "w0"), w0)

: // estimate regression model
: stata("quietly regress " + Ynm + " " + Znm + " if " + Dnm + "==0 [iw=w0]")
```

```
: Zg0      = Z * st_matrix("e(b)")'

: // compute IF for eta01
: h1       = X :* (D - p)

: G11inv   = invsym(cross(X, p :* (1 :- p), X) / N)

: h2       = Z :* w0 :* (Y :- Zg0)

: G21      = cross(-h2, X) / N

: G22inv   = invsym(cross(Z, w0, Z) / N)

: eta01    = mean(Zg0, D)

: h3       = D :* (Zg0 :- eta01)

: G32      = colsum(-D :* Z) / N

: IF_eta01 = N/sum(D) * (h3 - (h2 - h1 * G11inv' * G21') * G22inv' * G32')

: // compute IF for eta11
: eta11    = mean(Y, D)

: IF_eta11 = N/sum(D) * D :* (Y :- eta11)

: // compute IF for ATT
: ATT      = eta11 - eta01

: IF_ATT   = IF_eta11 - IF_eta01

: // display results (point estimate, mean of IF, standard error)
: (ATT, eta11, eta01)', mean((IF_ATT, IF_eta11, IF_eta01))',
>     sqrt(diagonal(variance((IF_ATT, IF_eta11, IF_eta01)) / N)) * sqrt((N-1)/N)
                1              2              3

  1 |  -.5761314967    2.97997e-10    1.252534622
  2 |   24.77272727   -6.54131e-16    1.377102927
  3 |   25.34885877   -2.97997e-10    .9753013655

: end
```

The results from `teffects` are identical:

```
. teffects ipwra (mpg price weight turn) (foreign price weight), atet

Iteration 0:   EE criterion =  8.949e-23
Iteration 1:   EE criterion =  5.694e-29

Treatment-effects estimation                    Number of obs     =         74
Estimator      : IPW regression adjustment
Outcome model  : linear
Treatment model: logit
```

mpg	Coef.	Robust Std. Err.	z	P>|z|	[95% Conf.	Interval]
ATET						

foreign (Foreign vs Domestic)	-.5761315	1.252535	-0.46	0.646	-3.031054	1.878791
POmean foreign						
Domestic	25.34886	.9753014	25.99	0.000	23.4373	27.26041

Average treatment effect on the untreated (ATC)

For the ATC, we have to flip some things around. The GMM problem is

$$\mathbf{h}_i(\mathbf{X}_i;\theta) = \begin{bmatrix} h_{i1}(x_i;\beta) \\ h_{i2}(z_i;\gamma_1) \\ h_{i3}(1;\eta_0^1) \end{bmatrix} = \begin{bmatrix} x_i'(d_i - p_i) \\ z_i'\omega_i^1(y_i - z_i\gamma_1) \\ (1-d_i)(z_i\gamma_1 - \eta_0^1) \end{bmatrix}$$

with

$$\hat{\omega}_i^1 = d_i\frac{1-\hat{p}_i}{\hat{p}_i} \qquad \text{and} \qquad \hat{p}_i = \frac{\exp(x_i\hat{\beta})}{1+\exp(x_i\hat{\beta})}$$

such that

$$\lambda_i(\hat{\eta}_0^1) = \frac{N}{\sum_i(1-d_i)}\left((1-d_i)(z_i\hat{\gamma}_1 - \hat{\eta}_0^1) - \overline{\mathbf{G}}_{32}\overline{\mathbf{G}}_{22}^{-1}\left(z_i'\hat{\omega}_i^1(y_i - z_i\hat{\gamma}_1) - \overline{\mathbf{G}}_{21}\overline{\mathbf{G}}_{11}^{-1}x_i'(d_i - \hat{p}_i)\right)\right) \tag{19}$$

with

$$\overline{\mathbf{G}}_{32} = \frac{1}{N}\sum_{i=1}^{N} -(1-d_i)z_i \qquad \overline{\mathbf{G}}_{21} = \frac{1}{N}\sum_{i=1}^{N} z_i'\hat{\omega}_i^1(y_i - z_i\hat{\gamma}_1)x_i$$

$$\overline{\mathbf{G}}_{22} = \frac{1}{N}\sum_{i=1}^{N} z_i'\hat{\omega}_i^1 z_i \qquad \overline{\mathbf{G}}_{11} = \frac{1}{N}\sum_{i=1}^{N} x_i'\hat{p}_i(1-\hat{p}_i)x_i$$

Example (reusing some results from above):

```
. mata:
------------------------------------------------- mata (type end to exit) -----
: // create new weights
: w1       = (1 :- p) :/ p :* D
: st_store(., st_addvar("double", "w1"), w1)
: // estimate regression model
```

```
: stata("quietly regress " + Ynm + " " + Znm + " if " + Dnm + "==1 [iw=w1]")

: Zg1      = Z * st_matrix("e(b)")'

: // compute IF for eta10
: h1       = X :* (D - p)

: G11inv   = invsym(cross(X, p :* (1 :- p), X) / N)

: h2       = Z :* w1 :* (Y :- Zg1)

: G21      = cross(h2, X) / N

: G22inv   = invsym(cross(Z, w1, Z) / N)

: eta10    = mean(Zg1, !D)

: h3       = !D :* (Zg1 :- eta10)

: G32      = colsum(- !D :* Z) / N

: IF_eta10 = N/sum(!D) * (h3 - (h2 - h1 * G11inv' * G21') * G22inv' * G32')

: // compute IF for eta00
: eta00    = mean(Y, !D)

: IF_eta00 = N/sum(!D) * !D :* (Y :- eta00)

: // compute IF for ATC
: ATC    = eta10 - eta00

: IF_ATC = IF_eta10 - IF_eta00

: // display results (point estimate, mean of IF, standard error)
: (ATC, eta10, eta00)', mean((IF_ATC, IF_eta10, IF_eta00))',
>     sqrt(diagonal(variance((IF_ATC, IF_eta10, IF_eta00)) / N)) * sqrt((N-1)/N)
                  1              2              3

  1 |  -13.18882902   -7.10303e-10    4.424288376
  2 |   6.638094059   -7.10302e-10    4.641527713
  3 |   19.82692308    2.80556e-16    .6514214963

: end
```

Confirm result using `teffects` (the treatment effect will have a wrong sign):

```
. teffects ipwra (mpg price weight turn) (foreign price weight), nolog atet ///
>     tlevel(0)

Treatment-effects estimation                    Number of obs     =         74
Estimator      : IPW regression adjustment
Outcome model  : linear
Treatment model: logit
```

mpg	Coef.	Robust Std. Err.	z	P>|z|	[95% Conf.	Interval]
ATET						
foreign						

(Domestic vs Foreign)	13.18883	4.424288	2.98	0.003	4.517383	21.86027
POmean foreign Foreign	6.638094	4.641528	1.43	0.153	-2.459133	15.73532

Average treatment effect (ATE)

Finally, the influence function for the ATE as usual (see page 23):

```
. mata:
------------------------------------------------- mata (type end to exit) -----
: p      = sum(D) / N
: ATE    = p * ATT + (1 - p) * ATC
: IF_ATE = p * IF_ATT :+ (1 - p) * IF_ATC :+ (ATT - ATC) * (D :- p)
: ATE, mean(IF_ATE), sqrt(variance(IF_ATE) / N) * sqrt((N-1)/N)
                 1              2              3
  1  -9.439108133   -4.10538e-10    3.169776549

: end
--------------------------------------------------------------------------------
```

```
. teffects ipwra (mpg price weight turn) (foreign price weight), nolog
Treatment-effects estimation                    Number of obs     =         74
Estimator      : IPW regression adjustment
Outcome model  : linear
Treatment model: logit
```

mpg	Coef.	Robust Std. Err.	z	P>|z|	[95% Conf.	Interval]
ATE foreign (Foreign vs Domestic)	-9.011602	3.937762	-2.29	0.022	-16.72947	-1.293731
POmean foreign Domestic	22.03224	.755597	29.16	0.000	20.5513	23.51318

As in the case of the IPW estimator, results from `teffects` are somewhat different because the ATE is computed differently. The estimator employed by `teffects` can be written as

$$\mathbf{h}_i(\mathbf{X}_i;\theta) = \begin{bmatrix} h_{i1}(x_i;\beta) \\ h_{i2}(z_i;\gamma_0) \\ h_{i3}(1;\eta^0) \\ h_{i4}(z_i;\gamma_1) \\ h_{i5}(1;\eta^1) \end{bmatrix} = \begin{bmatrix} x_i'(d_i - p_i) \\ z_i'\tilde{\omega}_i^0(y_i - z_i\gamma_0) \\ z_i\gamma_0 - \eta^0 \\ z_i'\tilde{\omega}_i^1(y_i - z_i\gamma_1) \\ z_i\gamma_1 - \eta^1 \end{bmatrix}$$

with $\tilde{\omega}_i^0 = (1-d_i)/(1-p_i)$ and $\tilde{\omega}_i^1 = d_i/p_i$ such that the influence functions are

$$\lambda_i(\hat{\eta}^0) = (z_i\hat{\gamma}_0 - \hat{\eta}^0) - \overline{\mathbf{G}}_{32}\overline{\mathbf{G}}_{22}^{-1}\left(z_i'\hat{\tilde{\omega}}_i^0(y_i - z_i\hat{\gamma}_0) - \overline{\mathbf{G}}_{21}\overline{\mathbf{G}}_{11}^{-1}x_i'(d_i - \hat{p}_i)\right) \tag{20}$$

with

$$\overline{\mathbf{G}}_{32} = \frac{1}{N}\sum_{i=1}^{N} -z_i \qquad \overline{\mathbf{G}}_{22} = \frac{1}{N}\sum_{i=1}^{N} z_i'\hat{\tilde{\omega}}_i^0 z_i \qquad \overline{\mathbf{G}}_{21} = \frac{1}{N}\sum_{i=1}^{N} -z_i\hat{\tilde{\omega}}_i^0(y_i - z_i\hat{\gamma}_0)\hat{p}_i x_i$$

and

$$\lambda_i(\hat{\eta}^1) = (z_i\hat{\gamma}_1 - \hat{\eta}^1) - \overline{\mathbf{G}}_{54}\overline{\mathbf{G}}_{44}^{-1}\left(z_i'\hat{\tilde{\omega}}_i^1(y_i - z_i\hat{\gamma}_1) - \overline{\mathbf{G}}_{41}\overline{\mathbf{G}}_{11}^{-1}x_i'(d_i - \hat{p}_i)\right) \tag{21}$$

with

$$\overline{\mathbf{G}}_{54} = \frac{1}{N}\sum_{i=1}^{N} -z_i \qquad \overline{\mathbf{G}}_{44} = \frac{1}{N}\sum_{i=1}^{N} z_i'\hat{\tilde{\omega}}_i^1 z_i \qquad \overline{\mathbf{G}}_{41} = \frac{1}{N}\sum_{i=1}^{N} z_i\hat{\tilde{\omega}}_i^1(y_i - z_i\hat{\gamma}_1)(1 - \hat{p}_i)x_i$$

and $\overline{\mathbf{G}}_{11}$ as above. Using these formulas, the ATE by `teffects` can be reproduced:

```
. sysuse auto, clear
(1978 Automobile Data)
. mata:
------------------------------------------------- mata (type end to exit) -----
: Dnm = "foreign"; Xnm = "price weight"
: Ynm = "mpg"    ; Znm = "price weight turn"
: N   = st_nobs()
: D   = st_data(., Dnm); X = st_data(., Xnm), J(N, 1, 1)
: Y   = st_data(., Ynm); Z = st_data(., Znm), J(N, 1, 1)
: // estimate logit and create weights
```

```
: stata("quietly logit " + Dnm + " " + Xnm)
: p        = invlogit(X * st_matrix("e(b)")')
: w0       = !D :/ (1 :- p)
: w1       =  D :/ p
: st_store(., st_addvar("double", "w0"), w0)
: st_store(., st_addvar("double", "w1"), w1)
: // estimate regression models
: stata("quietly regress " + Ynm + " " + Znm + " if " + Dnm + "==0 [iw=w0]")
: Zg0      = Z * st_matrix("e(b)")'
: stata("quietly regress " + Ynm + " " + Znm + " if " + Dnm + "==1 [iw=w1]")
: Zg1      = Z * st_matrix("e(b)")'
: // compute IF for eta0
: h1       = X :* (D - p)
: G11inv   = invsym(cross(X, p :* (1 :- p), X) / N)
: h2       = Z :* w0 :* (Y :- Zg0)
: G21      = cross(-h2, p, X) / N
: G22inv   = invsym(cross(Z, w0, Z) / N)
: eta0     = mean(Zg0)
: h3       = Zg0 :- eta0
: G32      = colsum(-Z) / N
: IF_eta0  = h3 - (h2 - h1 * G11inv' * G21') * G22inv' * G32'
: // compute IF for eta1
: h4       = Z :* w1 :* (Y :- Zg1)
: G41      = cross(h4, 1 :- p, X) / N
: G44inv   = invsym(cross(Z, w1, Z) / N)
: eta1     = mean(Zg1)
: h5       = Zg1 :- eta1
: G54      = colsum(-Z) / N
: IF_eta1  = h5 - (h4 - h1 * G11inv' * G41') * G44inv' * G54'
: // compute IF for ATT
: ATT      = eta1 - eta0
: IF_ATT   = IF_eta1 - IF_eta0
: // display results (point estimate, mean of IF, standard error)
: (ATT, eta1, eta0)', mean((IF_ATT, IF_eta1, IF_eta0))',
>     sqrt(diagonal(variance((IF_ATT, IF_eta1, IF_eta0)) / N)) * sqrt((N-1)/N)
                 1             2             3
  +-------------------------------------------+
1 |  -9.011601549   -6.78057e-10    3.937761567 |
2 |   13.02063846   -7.04676e-10    4.099591451 |
3 |   22.03224001   -2.66192e-11    .7555969982 |
  +-------------------------------------------+
```

```
: end
```

3.5 Exact matching

Exact matching stratifies the data based on unique patterns of the predictor values and then computes the outcome difference between treated and controls within each stratum (as long as the stratum contains observations from both groups). The treatment effect is then estimated as a stratum-size weighted average of the stratum-specific differences. In the following I will only focus on the ATT; results for the ATC (and, consequently, the ATE) can be obtained analogously.

Let J be the number of strata and let x_i be a $1 \times J$ indicator vector that identifies the stratum to which observation i belongs (that is, element j of x_i is equal to 1 if observations i belongs to stratum j, all other elements are 0). Exact matching can then be viewed as an inverse-probability weighting (IPW) estimator with a logit model of treatment status d_i on x_i (without constant), such that the GMM problem for η_1^0 is

$$\mathbf{h}_i(\mathbf{X}_i;\theta) = \begin{bmatrix} h_{i1}(x_i;\beta) \\ h_{i2}(1;\eta_1^0) \end{bmatrix} = \begin{bmatrix} x_i'(d_i - p_i) \\ \omega_i^0(y_i - \eta_1^0) \end{bmatrix}$$

where $p_i = \exp(x_i\beta)/(1+\exp(x_i\beta))$ and $\omega_i^0 = (1-d_i)\ p_i/(1-p_i)$. Hence

$$\overline{\mathbf{G}}(\hat{\theta}) = \frac{1}{N}\sum_{i=1}^{N} \begin{bmatrix} x_i'\hat{p}_i(1-\hat{p}_i)x_i & \mathbf{0} \\ -\hat{\omega}_i^0(y_i - \hat{\eta}_1^0)x_i & \hat{\omega}_i^0 \end{bmatrix}$$

Let Ω_i be the set of observations in observation i's stratum. Since x_i is an indicator vector, the influence function for η_1^0 can then be simplified to

$$\lambda_i(\hat{\eta}_1^0) = \frac{N}{\sum_i \hat{\omega}_i^0}\left(\hat{\omega}_i^0(y_i - \hat{\eta}_1^0) + \frac{\sum_{j\in\Omega_i}\hat{\omega}_j^0(y_j - \hat{\eta}_1^0)}{\sum_{j\in\Omega_i}\hat{p}_j(1-\hat{p}_j)}(d_i - \hat{p}_i)\right) \tag{22}$$

where $\hat{p}_i$ is the within stratum treatment probability. We see that the influence function has two components: a main component that is equal to the influence function of the weighted

mean of Y in the control group assuming the weights as fixed, plus an adjustment term accounting for the fact that the weights are estimated.

Alternatively, exact matching can also be viewed as a regression adjustment (RA) estimator based on a fixed-effects regression of y_i on indicator vector z_i (without constant; z_i is defined in the same way as x_i above), such that

$$\mathbf{h}_i(\mathbf{X}_i;\theta) = \begin{bmatrix} h_{i1}(z_i;\gamma_0) \\ h_{i2}(1;\eta_1^0) \end{bmatrix} = \begin{bmatrix} z_i'(1-d_i)(y_i - z_i\gamma_0) \\ d_i(z_i\gamma_0 - \eta_1^0) \end{bmatrix}$$

and, hence,

$$\overline{\mathbf{G}}(\hat{\theta}) = \frac{1}{N}\sum_{i=1}^{N} \begin{bmatrix} z_i'(1-d_i)z_i & \mathbf{0} \\ -d_i z_i & d_i \end{bmatrix}$$

Let $\hat{y}_i^0 = z_i\hat{\gamma}_0$ be the estimate of the the potential outcome without treatment for observation i (which is equal to the mean of Y among the controls in Ω_i). The influence function for η_1^0 can then be written as:

$$\lambda_i(\hat{\eta}_1^0) = \frac{N}{\sum_i d_i}\left(d_i(\hat{y}_i^0 - \hat{\eta}_1^0) + \frac{\sum_{j\in\Omega_i} d_j}{\sum_{j\in\Omega_i}(1-d_j)}(1-d_i)(y_i - \hat{y}_i^0) \right) \tag{23}$$

Again there are two components: a main component that is equal to the influence function we would get if we compute $\hat{\eta}_1^0$ as mean of $\hat{y}_i^0$ in the treatment group assuming $\hat{y}_i^0$ as fixed, and a correction term that adjusts for the fact that $\hat{y}_i^0$ is estimated.

Note that $t_i = \sum_{j\in\Omega_i} d_i$ is the number of treatment cases in observation i's stratum, and $c_i = \sum_{j\in\Omega_i}(1-d_i)$ is the number of control cases in the stratum. Furthermore, since $\sum_i \hat{\omega}_i^0 = \sum_i d_i$, $\hat{p}_i = t_i/(t_i+c_i)$, $\hat{\omega}_i^0 = 0$ if $d_i = 1$ and $\hat{\omega}_i^0 = t_i/c_i$ if $d_i = 0$, and, hence, $\sum_{j\in\Omega_i}\hat{p}_j(1-\hat{p}_j) = (t_i+c_i)\hat{p}_i(1-\hat{p}_i) = t_i c_i/(t_i+c_i)$ as well as $\sum_{j\in\Omega_i}\hat{\omega}_j^0(y_j - \hat{\eta}_1^0) = t_i(\hat{y}_i^0 - \hat{\eta}_1^0)$, we see that Equation 22 can be transformed into Equation 23 after some rearrangement.

That is, both approaches, the IPW representation and the RA representation of exact matching, lead to the same result. In terms of computation, however, the regression-adjustment representation (Equation 23) is somewhat more convenient. Example:

```
. sysuse nlsw88, clear
(NLSW, 1988 extract)
```

```
. keep if union<. & grade<. & race<. & wage<.
(370 observations deleted)

. // genedate a stratum id
. sort grade race

. by grade race: generate sid = (_n==1)

. replace sid = sum(sid)
(1,875 real changes made)

. // count number of observations within stratum
. sort sid

. by sid: generate T = sum(union)

. by sid: replace  T = T[_N]      // size of treatment group within stratum
(1755 real changes made)

. by sid: generate C = sum(union==0)

. by sid: replace  C = C[_N]      // size of control group within stratum
(1823 real changes made)

. // exclude strata that have insufficient treatment variability (with default
. // settings, teffect requires at least 3 treated and 3 controls per stratum)
. keep if T>=3 & C>=3
(44 observations deleted)

. // compute within-stratum outcome averages in control group
. by sid: generate double y0 = sum(wage*(union==0))

. by sid: replace         y0 = y0[_N] / C
(1832 real changes made)

. mata:
------------------------------------------------- mata (type end to exit) -----
: N   = st_nobs()

: D   = st_data(.,"union")

: Y   = st_data(.,"wage")

: T   = st_data(.,"T")

: C   = st_data(.,"C")

: y0  = st_data(.,"y0")

: // compute IF for eta01
: eta01    = mean(y0, D)

: IF_eta01 = N/sum(D) * (D :* (y0 :- eta01) + T :/ C :* !D :* (Y - y0))

: // compute IF for eta11
: eta11    = mean(Y, D)

: IF_eta11 = N/sum(D) * D :* (Y :- eta11)

: // compute IF for ATT
: ATT      = eta11 - eta01

: IF_ATT   = IF_eta11 - IF_eta01

: // display results (point estimate, mean of IF, standard error)
: (ATT, eta11, eta01)', mean((IF_ATT, IF_eta11, IF_eta01))',
```

```
>     sqrt(diagonal(variance((IF_ATT, IF_eta11, IF_eta01)) / N))
                  1             2             3

  1    1.177104006   2.43570e-16   .2221082085
  2    8.729361226   7.14933e-16   .1970700505
  3     7.55225722   4.58804e-16   .1543704822

: end
```

The resulting standard error for the ATT is very close, at least in this example, to what `teffects` reports, even though `teffects` uses a different estimation methodology (which is based on Abadie and Imbens, 2006; see [TE] **teffects nnmatch**):

```
. teffects nnmatch (wage) (union), ematch(grade race) atet

Treatment-effects estimation                   Number of obs      =      1,832
Estimator      : nearest-neighbor matching     Matches: requested =          1
Outcome model  : matching                                     min =          3
Distance metric: Mahalanobis                                  max =        453
```

wage	Coef.	AI Robust Std. Err.	z	P>|z|	[95% Conf.	Interval]
ATET						
union (union vs nonunion)	1.177104	.2216018	5.31	0.000	.7427725	1.611435

3.6 Mahalanobis distance matching

Think of Mahalanobis distance matching as a GMM estimator similar to the regression-adjustment representation of exact matching. Order the observations such that $i = 1, \ldots, t$ indexes the observations in the treatment group and $i = t + 1, \ldots, t + c$ indexes the observations in the control group, where t is the size of the treatment group and c is the size of the control group. Matching produces a potential outcome estimate $\hat{y}^0_j$, $j = 1, \ldots, t$, for each observation in the treatment group by taking a weighted average of the outcomes in the control group, where weight ω^0_{ji} denotes the weight that control i receives in the computation of $\hat{y}^0_j$ (the weight will be zero for observations in the treatment group and also for observa-

tions in the control group, that have not been used as matches for treatment observation j). Assuming the matching configuration (i.e. the weights ω_{ij}^0) as fixed, the moment equations of the GMM problem for η_1^0 can then be written as

$$\mathbf{h}_i(\mathbf{1};\theta) = \begin{bmatrix} h_{i1}(1;y_1^0) \\ h_{i2}(1;y_2^0) \\ \vdots \\ h_{it}(1;y_t^0) \\ h_{i(t+1)}(1;\eta_1^0) \end{bmatrix} = \begin{bmatrix} \omega_{i1}^0(y_i - y_1^0) \\ \omega_{i2}^0(y_i - y_2^0) \\ \vdots \\ \omega_{it}^0(y_i - y_t^0) \\ d_i\left(\sum_{j=1}^t y_j^0 1_{\{i=j\}} - \eta_1^0\right) \end{bmatrix}$$

where $1_{\{a\}}$ is the indicator function (equal to 1 if a is true and 0 else), such that

$$\overline{\mathbf{G}}(\hat{\theta}) = \frac{1}{N}\sum_{i=1}^N \begin{bmatrix} \omega_{i1}^0 & 0 & \dots & 0 & 0 \\ 0 & \omega_{i2}^0 & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & \omega_{it}^0 & 0 \\ -d_i 1_{\{i=1\}} & -d_i 1_{\{i=2\}} & \dots & -d_i 1_{\{i=t\}} & d_i \end{bmatrix}$$

Assume that the matching weights ω_{ij}^0 are normalized such that $\sum_i \omega_{ij}^0 = 1$. The influence function for η_1^0 can then be written as

$$\begin{aligned} \lambda_i(\hat{\eta}_1^0) &= \frac{N}{\sum_i d_i}\left(d_i(\hat{y}_i^0 - \hat{\eta}_1^0) + \sum_{j=1}^t \omega_{ij}^0(y_i - \hat{y}_j^0)\right) \\ &= \frac{N}{\sum_i d_i}\left(d_i(\hat{y}_i^0 - \hat{\eta}_1^0) + \omega_i^0\left(y_i - \frac{1}{\omega_i^0}\sum_{j=1}^t \omega_{ij}^0 \hat{y}_j^0\right)\right) \end{aligned} \tag{24}$$

where $\omega_i^0 = \sum_{j=1}^t \omega_{ij}^0$ is the overall matching weight of observation i. Example:[6]

```
. clear all
. local ss string scalar
```

[6]The example uses 3-nearest-neighbor matching (with replacement), and the data has no ties so that each treatment observations is matched to exactly 3 controls. The weights ω_{ij} are therefore always equal to 1/3 (or zero). For situations in which the number of matches varies, for example because there are ties, or if using radius or kernel matching, the code would have to be adapted.

```
. mata:
------------------------------------------------------------------------------- mata (type end to exit) -------
: void IF_ATT(`ss' Dnm, `ss' Ynm)
> {
>     N       = st_nobs()
>     D       = st_data(., Dnm)
>     Y       = st_data(., Ynm)
>     y0      = st_data(., "PO")
>     IDs     = st_data(., "ID1 ID2 ID3")
>     ATTte   = st_matrix("e(b)")
>     V_ATTte = st_matrix("e(V)")
>     // compute IF for eta11
>     wr = J(N,1,0) // sum_j w_ij (y_i - y0_j)
>     for (i=1; i<=N; i++) {
>         if (D[i]) continue
>         for (j=1; j<=N; j++) {
>             if (!D[j]) continue
>             if (anyof(IDs[j,],i)) {
>                 wi = 1/cols(IDs) // weight of control i with respect to j
>                 wr[i] = wr[i] + wi * (Y[i] - y0[j])
>             }
>         }
>     }
>     eta01    = mean(y0, D)
>     IF_eta01 = N/sum(D) * (D :* (y0 :- eta01) + wr)
>     // compute IF for eta11
>     eta11    = mean(Y, D)
>     IF_eta11 = N/sum(D) * D :* (Y :- eta11)
>     // compute IF for ATT
>     ATT      = eta11 - eta01
>     IF_ATT   = IF_eta11 - IF_eta01
>     assert (reldif(ATT, ATTte)<1e-12)
>     // Results
>     b = (ATT, eta11, eta01)
>     V = variance((IF_ATT, IF_eta11, IF_eta01))/N
>     b', mean((IF_ATT, IF_eta11, IF_eta01))', sqrt(diagonal(V))
>     b = (ATTte, b)
>     V = blockdiag(V_ATTte, V)
>     st_rclear()
>     st_matrix("b", b)
>     st_matrix("V", V)
>     cstripe = (J(1,4,"") \ ("ATTte", "ATT", "eta11", "eta01"))'
>     st_matrixcolstripe("b", cstripe)
>     st_matrixcolstripe("V", cstripe)
>     st_matrixrowstripe("V", cstripe)
> }

: end
------------------------------------------------------------------------------------------------------------

. sysuse auto, clear
(1978 Automobile Data)

. teffects nnmatch (mpg price weight) (foreign), atet nn(3) generate(ID*)
```

```
Treatment-effects estimation                   Number of obs      =         74
Estimator      : nearest-neighbor matching     Matches: requested =          3
Outcome model  : matching                                  min =          3
Distance metric: Mahalanobis                               max =          3
```

mpg	Coef.	AI Robust Std. Err.	z	P>|z|	[95% Conf.	Interval]
ATET						
foreign (Foreign vs Domestic)	-.969697	1.50149	-0.65	0.518	-3.912564	1.97317

```
. predict double PO, po

. mata: IF_ATT("foreign", "mpg")
                 1              2              3
  +-------------------------------------------------+
1 |  -.9696969697    2.84457e-15     1.37343757  |
2 |   24.77272727   -6.54131e-16    1.386503056  |
3 |   25.74242424   -3.44169e-15    1.234977925  |
  +-------------------------------------------------+
```

Results are quite different from what `teffects` returns, but this is not overly surprising since `teffects` uses a different methodology to compute the standard errors and the sample is very small. To see whether our approach provides valid results, we can run some simulations:

```
. program mysim, eclass
  1.     syntax [, n(int 100) ]
  2.     drop _all
  3.     set obs `n'
  4.     matrix C = (1, .5, 1)
  5.     drawnorm X1 X2, corr(C) cstorage(lower) double
  6.     gen byte D = ((X1 + X2)/10 + rnormal())>.3
  7.     gen double Y = 1 + .2*X1 + .3*X2 + D*(.5 + .4*X1 + .3*X2) + ///
>         rnormal(0, exp(0 + .2*X1))
  8.     teffects nnmatch (Y X1 X2) (D), atet nn(3) generate(ID*)
  9.     predict double PO, po
 10.     mata: IF_ATT("D", "Y")
 11.     eret post b V
 12.     eret local cmd "mysim"
 13. end

. simulate _b _se, reps(10000) nodots nolegend: mysim

. su _b*
```

Variable	Obs	Mean	Std. Dev.	Min	Max
_b_ATTte	10,000	.6214599	.2722596	-.4402424	1.739156
_b_ATT	10,000	.6214599	.2722596	-.4402424	1.739156

```
    _b_eta11 |      10,000    1.674874    .2402745    .7907771    2.581279
    _b_eta01 |      10,000    1.053414    .1882419    .3131391    1.996144

. su _se*

    Variable |        Obs        Mean    Std. Dev.       Min        Max
-------------+---------------------------------------------------------
   _se_ATTte |      10,000    .2674574    .0339971    .1697307    .4596234
     _se_ATT |      10,000     .261992    .0337009    .1664717    .4178433
   _se_eta11 |      10,000    .2372465     .033139    .1418522    .4035607
   _se_eta01 |      10,000    .1749407    .0284664     .104253    .3978903

. corr _se_ATTte _se_ATT
(obs=10,000)

             | _se_AT~e  _se_ATT
-------------+------------------
   _se_ATTte |   1.0000
     _se_ATT |   0.9489   1.0000
```

This might be an idealized example (no ties, well-behaved continuous data), but the results do not look too bad. The standard errors based on Equation 24 capture the sampling variance of the ATT and $\hat{\eta}_1^0$ pretty well, despite the small sample size of $N = 100$. However, there is still some bias: For the ATT, the standard deviation across simulations is .27226, the average of the standard error estimate is .261992 (bias of -3.77 percent); for $\hat{\eta}_1^0$, the observed standard deviation is .188242 and the average standard error is .174941 (bias of -7.07 percent). The performance of `teffetcs` appears to be a bit better (the bias in the standard error estimate for the ATT from `teffetcs` is -1.76 percent).

The reason for the remaining bias in Equation 24 is that the matching configuration is treated as fixed. If the X variables are random, some additional uncertainty will occur because the matching solution will vary from sample to sample. Such variability in the matching solution will result in additional variability of $\hat{y}_i^0$ if there is variance in the Y values among potential matches. Possibly, standard error estimation could be improved by using a second matching algorithm to look for potential replacement matches in the local neighborhood of a control (by matching controls on controls) and then somehow incorporate the variability of these Y values into the influence function (in the spirit of the procedure by Abadie and Imbens, 2006), but I leave this open for further research.

3.7 Bias-adjusted Mahalanobis distance matching

Bias-adjusted Mahalanobis distance matching is like Mahalanobis distance matching followed by regression adjustment. Again treating the matching configuration (i.e. the weights ω_{ji}^0) as fixed, the GMM problem can be written as

$$\mathbf{h}_i(\mathbf{1};\theta) = \begin{bmatrix} h_{i0}(z_i;\gamma_0) \\ h_{i1}(1;y_1^0) \\ h_{i2}(1;y_2^0) \\ \vdots \\ h_{it}(1;y_t^0) \\ h_{i(t+1)}(1;\eta_1^0) \end{bmatrix} = \begin{bmatrix} \sum_{j=1}^t \omega_{ji}^0(y_i - z_i\gamma_0) \\ \omega_{i1}^0(y_i + (z_1 - z_i)\gamma_0 - y_1^0) \\ \omega_{i2}^0(y_i + (z_2 - z_i)\gamma_0 - y_2^0) \\ \vdots \\ \omega_{it}^0(y_i + (z_t - z_i)\gamma_0 - y_t^0) \\ d_i\left(\sum_{j=1}^t y_j^0 1_{\{i=j\}} - \eta_1^0\right) \end{bmatrix}$$

such that

$$\overline{\mathbf{G}}(\hat{\theta}) = \frac{1}{N}\sum_{i=1}^N \begin{bmatrix} z_i'\sum_{j=1}^t \omega_{ji}^0 z_i & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ -\omega_{i1}^0(z_1 - z_i) & \omega_{i1}^0 & 0 & \dots & 0 & 0 \\ -\omega_{i2}^0(z_2 - z_i) & 0 & \omega_{i2}^0 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ -\omega_{it}^0(z_t - z_i) & 0 & 0 & \dots & \omega_{it}^0 & 0 \\ \mathbf{0} & -d_i 1_{\{i=1\}} & -d_i 1_{\{i=2\}} & \dots & -d_i 1_{\{i=t\}} & d_i \end{bmatrix}$$

Note that $\sum_i \omega_{ij}^0 = 1$ and let

$$\overline{\mathbf{G}}_{11} = \frac{1}{N}\sum_{i=1}^N z_i' \sum_{j=1}^t \omega_{ij}^0 z_i = \frac{1}{N}\sum_{i=1}^N z_i'\omega_i^0 z_i$$

$$\overline{\mathbf{G}}_{(1+j)1} = \frac{1}{N}\sum_{i=1}^N -\omega_{ij}^0(z_j - z_i) = -\frac{1}{N}(z_j - \bar{z}_j), \quad j = 1,\dots,t$$

where $\omega_i^0 = \sum_j \omega_{ij}^0$ and $\bar{z}_j^0 = \sum_{i=1}^N \omega_{ij}^0 z_i$. The influence function can then be written as

$$\lambda_i(\hat{\eta}_1^0) = \frac{N}{\sum_i d_i}\left(d_i(\hat{y}_i^0 - \hat{\eta}_1^0) + \sum_{j=1}^t \left(\omega_{ij}^0(y_i + (z_j - z_i)\hat{\gamma}_0 - \hat{y}_j^0) + \frac{(z_j - \bar{z}_j^0)}{N}\lambda_i(\hat{\gamma}_0)\right)\right) \quad (25)$$

with

$$\lambda_i(\hat{\gamma}_0) = \overline{\mathbf{G}}_{11}^{-1} z_i' \sum_j \omega_{ij}^0(y_i - z_i\hat{\gamma}_0) = \overline{\mathbf{G}}_{11}^{-1} z_i'\omega_i^0(y_i - z_i\hat{\gamma}_0)$$

and $\hat{y}_j^0 = \sum_i \omega_{ij}^0 (y_i + (z_j - z_i)\hat{\beta}^0)$. Example:

```
. clear all
. local ss string scalar
. mata:
------------------------------------------------- mata (type end to exit) -------
: void IF_ATT(`ss' Dnm, `ss' Ynm, `ss' Znm)
> {
>     N       = st_nobs()
>     D       = st_data(., Dnm)
>     Y       = st_data(., Ynm)
>     IDs     = st_data(., "ID1 ID2 ID3")
>     Z       = st_data(.,Znm), J(N,1,1)
>     ATTte   = st_matrix("e(b)")
>     V_ATTte = st_matrix("e(V)")
>     // obtain matching weights
>     w0 = J(N,1,0)
>     for (i=1; i<=N; i++) {
>         if (D[i]) continue
>         for (j=1; j<=N; j++) {
>              if (!D[j]) continue
>              if (anyof(IDs[j,],i)) {
>                  wi = 1/cols(IDs) // weight of control i with respect to j
>                  w0[i] =  w0[i] + wi
>              }
>         }
>     }
>     // compute RA coefficients
>     st_store(., st_addvar("double", "w0"), w0)
>     stata("quietly regress " + Ynm + " " + Znm + " [aw=w0]")
>     g0 = st_matrix("e(b)")'
>     // compute IF_g0
>     G11inv = invsym(cross(Z, w0, Z) / N)
>     IF_g0  = w0 :* (Y :- Z*g0) :* Z * G11inv'
>     // compute y0 and z0
>     y0 = J(N,1,0)
>     z0 = J(N,cols(Z),0)
>     for (j=1; j<=N; j++) {
>         if (!D[j]) continue
>         y0[j]  = sum(Y[IDs[j,]] + (Z[j,] :- Z[IDs[j,],])*g0) / cols(IDs)
>         z0[j,] = colsum(Z[IDs[j,],]) / cols(IDs)
>     }
>     // compute correction term
>     wr = J(N,1,0)
>     for (i=1; i<=N; i++) {
>         if (D[i]) continue
>         for (j=1; j<=N; j++) {
>             if (!D[j]) continue
>             if (anyof(IDs[j,],i)) {
>                 wi = 1/cols(IDs) // weight of control i with respect to j
>                 wr[i] = wr[i] + wi * (Y[i] + (Z[j,] - Z[i,])*g0 - y0[j])
>             }
```

```
>             wr[i] = wr[i] + IF_g0[i,] * (Z[j,] - z0[j,])'/N
>         }
>     }
>     // compute IF for eta11
>     eta01    = mean(y0, D)
>     IF_eta01 = N/sum(D) * (D :* (y0 :- eta01) + wr)
>     // compute IF for eta11
>     eta11    = mean(Y, D)
>     IF_eta11 = N/sum(D) * D :* (Y :- eta11)
>     // compute IF for ATT
>     ATT      = eta11 - eta01
>     IF_ATT   = IF_eta11 - IF_eta01
>     assert (reldif(ATT, ATTte)<1e-12)
>     // Results
>     b = (ATT, eta11, eta01)
>     V = variance((IF_ATT, IF_eta11, IF_eta01))/N
>     b', mean((IF_ATT, IF_eta11, IF_eta01))', sqrt(diagonal(V))
>     b = (ATTte, b)
>     V = blockdiag(V_ATTte, V)
>     st_rclear()
>     st_matrix("b", b)
>     st_matrix("V", V)
>     cstripe = (J(1,4,"") \ ("ATTte", "ATT", "eta11", "eta01"))'
>     st_matrixcolstripe("b", cstripe)
>     st_matrixcolstripe("V", cstripe)
>     st_matrixrowstripe("V", cstripe)
> }

: end
```

```
. sysuse auto, clear
(1978 Automobile Data)

. teffects nnmatch (mpg price weight) (foreign), atet nn(3) ///
>     biasadj(price weight) generate(ID*)

Treatment-effects estimation                   Number of obs      =         74
Estimator      : nearest-neighbor matching     Matches: requested =          3
Outcome model  : matching                                 min =          3
Distance metric: Mahalanobis                              max =          3
```

mpg	AI Robust Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
ATET						
foreign (Foreign vs Domestic)	-2.057838	1.48609	-1.38	0.166	-4.970521	.8548451

```
. mata: IF_ATT("foreign", "mpg", "price weight")
                 1             2             3
  +-------------------------------------------+
1 |  -2.057838086  -1.76736e-15   1.568344364  |
  +-------------------------------------------+
```

```
  2 |   24.77272727   -6.54131e-16    1.386503056
  3 |   26.83056536    1.13273e-15    1.284358244
```

The resulting standard error for the ATT is in a similar range as the standard error returned by `teffects`. To evaluate the estimator we can again run some simulations:

```
. program mysim, eclass
  1.     syntax [, n(int 100) ]
  2.     drop _all
  3.     set obs `n'
  4.     matrix C = (1, .5, 1)
  5.     drawnorm X1 X2, corr(C) cstorage(lower) double
  6.     gen byte D = ((X1 + X2)/10 + rnormal())>.3
  7.     gen double Y = 1 + .2*X1 + .3*X2 + D*(.5 + .4*X1 + .3*X2) + ///
>         rnormal(0, exp(0 + .2*X1))
  8.     teffects nnmatch (Y X1 X2) (D), atet nn(3) biasadj(X1 X2) generate(ID*)
  9.     mata: IF_ATT("D", "Y", "X1 X2")
 10.     eret post b V
 11.     eret local cmd "mysim"
 12. end

. simulate _b _se, reps(10000) nodots nolegend: mysim

. su _b*

    Variable |        Obs        Mean    Std. Dev.       Min        Max
-------------+---------------------------------------------------------
    _b_ATTte |     10,000     .601353    .2707446  -.4625704   1.730218
      _b_ATT |     10,000     .601353    .2707446  -.4625704   1.730218
    _b_eta11 |     10,000    1.674874    .2402745   .7907771   2.581279
    _b_eta01 |     10,000    1.073521    .1966577   .3090912     2.0711

. su _se*

    Variable |        Obs        Mean    Std. Dev.       Min        Max
-------------+---------------------------------------------------------
   _se_ATTte |     10,000    .2635178    .0340952   .1667601   .4544649
     _se_ATT |     10,000     .260089    .0351009   .1623164   .4307056
   _se_eta11 |     10,000    .2372465     .033139   .1418522   .4035607
   _se_eta01 |     10,000    .1830322    .0320217   .1047215   .4163064

. corr _se_ATTte _se_ATT
(obs=10,000)

             | _se_AT~e  _se_ATT
-------------+------------------
   _se_ATTte |   1.0000
     _se_ATT |   0.9450   1.0000
```

Results are similar to matching without bias adjustment. For the ATT, the standard error estimate based on Equation 25 has a bias of -3.94 percent; for the standard error of $\hat{\eta}_1^0$

the bias is −6.93 percent. The performance of `teffetcs` is somewhat better with a bias of −2.67 percent for the ATT.

3.8 Entropy balancing

Basic model

The original formal exposition on entropy balancing in Hainmueller (2012) is difficult to understand, at least in my opinion. In essence, however, entropy balancing is straightforward. It looks for an estimate of $\theta = (\alpha, \beta')'$ such that

$$\frac{1}{\sum_{i=1}^{N} \omega_i} \sum_{i=1}^{N} \omega_i x_i' = \mu \qquad \text{and} \qquad \sum_{i=1}^{N} \omega_i = t \qquad \text{with} \qquad \omega_i = \exp(\alpha + x_i\beta)$$

where μ is a $(k-1) \times 1$ vector of target means, one for each variable, and t is a target sum of weights. In terms of a GMM estimator, entropy balancing can be expressed as

$$\mathbf{h}_i(\mathbf{X}_i; \theta) = \begin{bmatrix} h_{i1}(1; \alpha) \\ h_{i2}(x_i; \beta) \end{bmatrix} = \begin{bmatrix} \omega_i - \frac{t}{N} \\ \omega_i(x_i' - \mu) \end{bmatrix}$$

where the first moment equation ensures that the sum of ω_i is equal to t. The influence function for $\hat{\theta}$ thus is

$$\lambda_i^{\hat{\theta}} = \overline{\mathbf{G}}(\hat{\theta})^{-1} \mathbf{h}_i(\mathbf{X}_i; \hat{\theta})$$

with

$$\overline{\mathbf{G}}(\hat{\theta}) = \frac{1}{N} \sum_{i=1}^{N} \begin{bmatrix} -\hat{\omega}_i & -\hat{\omega}_i x_i \\ h_{i2}(x_i; \hat{\beta}) & -h_{i2}(x_i; \hat{\beta}) x_i \end{bmatrix}$$

where

$$\hat{\omega}_i = \exp(\hat{\alpha} + x_i\hat{\beta}) = \exp(\hat{\alpha}) \exp(x_i\hat{\beta})$$

Note that $\overline{\mathbf{G}}_{21} = \mathbf{0}$ because $\sum h_{i2}(x_i; \beta)$ is zero by definition (at least if the model converges). Demonstration of entropy balancing using GMM:

```
. sysuse auto, clear
(1978 Automobile Data)

. // get target mean for price
. su price if foreign==1
```

```
    Variable |        Obs        Mean    Std. Dev.       Min        Max
-------------+---------------------------------------------------------
       price |         22    6384.682    2621.915       3748      12990

. local m1 = r(mean)

. // get target mean for weight
. su weight if foreign==1

    Variable |        Obs        Mean    Std. Dev.       Min        Max
-------------+---------------------------------------------------------
      weight |         22    2315.909    433.0035       1760       3420

. local m2 = r(mean)

. // get target sum of weights
. quietly count if foreign==1

. local T  = r(N)

. // get sample size (subsample to which entropy balancing will be applied)
. quietly count if foreign==0

. local N = r(N)

. // run GMM
. local w "exp({a} + {b1}*price + {b2}*weight)"

. gmm (`w' - `T'/`N')         ///
>     (`w'*(price  - `m1'))   ///
>     (`w'*(weight - `m2'))   ///
>     if foreign==0, winitial(identity) nolog

Final GMM criterion Q(b) =  2.91e-31

note: model is exactly identified

GMM estimation

Number of parameters =   3
Number of moments    =   3
Initial weight matrix: Identity                   Number of obs   =         52
GMM weight matrix:     Robust

------------------------------------------------------------------------------
             |               Robust
             |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
          /a |   7.065282   1.978246     3.57   0.000     3.187992    10.94257
         /b1 |    .000972    .000231     4.21   0.000     .0005193    .0014246
         /b2 |  -.0052477   .0012756    -4.11   0.000    -.0077478   -.0027477
------------------------------------------------------------------------------

Instruments for equation 1: _cons
Instruments for equation 2: _cons
Instruments for equation 3: _cons

. // confirm that resulting weights do what they are supposed to do
. generate double w = exp(_b[/a] + _b[/b1]*price + _b[/b2]*weight) if e(sample)
(22 missing values generated)

. su price weight [aw=w] if foreign==0
```

```
    Variable |      Obs      Weight        Mean   Std. Dev.       Min        Max
-------------+---------------------------------------------------------------------
       price |       52          22    6384.682    4013.89        3291       15906
      weight |       52          22    2315.909   846.6718        1800        4840

. // computation if IFs
. gen byte domestic = (1-foreign)

. mata:
------------------------------------------------- mata (type end to exit) -------
: m = (`m1', `m2')

: k = length(m)

: a = st_matrix("e(b)")[,1]

: b = st_matrix("e(b)")[,(2,3)]'

: N = `N'; T = `T'

: X = st_data(., "price weight", "domestic")

: // prepare weights and h(x,b)
: w = exp(a :+ X*b)

: ha = (w :- T/N)

: hb = w :* (X :- m)

: // compute G
: Ginv = luinv((-sum(w)    , -colsum(w :* X) \
>               colsum(hb)', -cross(X,hb)    ) / N)

: // compute IF
: IF = (ha, hb) * Ginv'

: // display results (point estimate, mean of IF, standard error)
: (a, b')', mean((IF))', sqrt(diagonal(variance(IF) / N)) * sqrt((N-1)/N)
                 1              2              3
  +-------------------------------------------------------------+
1 |     7.0652823    1.08033e-15    1.978245763                 |
2 |    .0009719645   -1.02605e-19    .0002309577                |
3 |   -.0052477389   -5.89014e-20    .0012755569                |
  +-------------------------------------------------------------+

: end
-------------------------------------------------------------------------------
```

Average treatment effect on the treated (ATT)

In causal inference we are typically interested in the standard error of a reweighted outcome mean, but not the entropy balancing coefficients per se. For the estimation of an ATT, define the entropy balancing weights as

$$\omega_i^0 = (1 - d_i) \exp(\alpha + x_i\beta)$$

such that

$$\eta_1^0 = \frac{1}{\sum_{i=1}^N \omega_i^0} \sum_{i=1}^N \omega_i^0 y_i$$

where y_i is an outcome variable of interest. Furthermore, assume that the target means μ are estimated from the observations in the treatment group and that the target sum of weights is equal to the size of the treatment group, that is, $t = pN$, where p is the treatment probability.[7] Our interest lies in computing influence function for η_1^0 (as usual, the influence functions for η_1^1 and the ATT can be computed separately in a second step). The GMM problem for η_1^0 can be expressed as

$$\mathbf{h}_i(\mathbf{X}_i; \theta) = \begin{bmatrix} h_{i1}(x_i; \mu) \\ h_{i2}(1; p) \\ h_{i3}(1; \alpha) \\ h_{i4}(x_i; \beta) \\ h_{i5}(1; \eta_1^0) \end{bmatrix} = \begin{bmatrix} d_i(x_i' - \mu) \\ d_i - p \\ \omega_i^0 - \frac{(1-d_i)p}{1-p} \\ \omega_i^0(x_i' - \mu) \\ \omega_i^0(y_i - \eta_1^0) \end{bmatrix}$$

with

$$\overline{\mathbf{G}}(\hat{\theta}) = \frac{1}{N} \sum_{i=1}^N \begin{bmatrix} d_i I_k & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & 1 & 0 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \frac{1-d_i}{(1-\hat{p})^2} & -\hat{\omega}_i^0 & -\hat{\omega}_i^0 x_i & \mathbf{0} \\ \hat{\omega}_i I_k & \mathbf{0} & h_{i4}(x_i; \hat{\beta}) & -h_{i4}(x_i; \hat{\beta}) x_i & \mathbf{0} \\ \mathbf{0} & 0 & h_{i5}(1; \hat{\eta}_1^0) & -h_{i5}(1; \hat{\eta}_1^0) x_i & \hat{\omega}_i^0 \end{bmatrix}$$

where I_k is the identity matrix of size k. Since $\sum h_{i4}(x_i; \hat{\beta}) = \mathbf{0}$ and $\sum h_{i5}(1; \hat{\eta}_1^0) = \mathbf{0}$ by definition, the influence function of $\hat{\eta}_1^0$ can be written as

$$\lambda_i(\hat{\eta}_1^0) = \frac{N}{\sum_i \hat{\omega}_i} \left(h_{i5}(1; \hat{\eta}_1^0) - \overline{\mathbf{G}}_{54} \overline{\mathbf{G}}_{44}^{-1} \left(h_{i4}(x_i; \hat{\beta}) - \overline{\mathbf{G}}_{41} \overline{\mathbf{G}}_{11}^{-1} h_{i1}(x_i; \hat{\mu}) \right) \right) \tag{26}$$

with

$$\overline{\mathbf{G}}_{54} = \frac{1}{N} \sum_{i=1}^N -h_{i5}(1; \hat{\eta}_1^0) x_i \qquad\qquad \overline{\mathbf{G}}_{41} = \frac{1}{N} \sum_{i=1}^N \hat{\omega}_i^0 I_k$$

[7] Be aware that entropy balancing can also balance higher moments or even covariances between predictors. I only focus on means here to keep the discussion simple. The extension to higher moments and covariances is straightforward as these are just moment conditions on powers and interactions of the predictors.

$$\overline{\mathbf{G}}_{44} = \frac{1}{N}\sum_{i=1}^{N} -h_{i4}(x_i;\hat{\beta})x_i \qquad\qquad \overline{\mathbf{G}}_{11} = \frac{1}{N}\sum_{i=1}^{N} d_i I_k$$

The computation goes as follows:

```
. sysuse auto, clear
(1978 Automobile Data)

. local w "(foreign==0) * exp({a} + {b1}*price + {b2}*weight)"

. gmm (foreign*(price  - {m1}))           ///
>     (foreign*(weight - {m2}))           ///
>     (foreign - {p})                     ///
>     (`w' - (foreign==0)*{p}/(1-{p}))    ///
>     (`w'*(price  - {m1}))               ///
>     (`w'*(weight - {m2}))               ///
>     (`w'*(mpg    - {eta01}))            ///
>     , winitial(identity) nolog

Final GMM criterion Q(b) =  2.74e-31

note: model is exactly identified

GMM estimation

Number of parameters =   7
Number of moments    =   7
Initial weight matrix: Identity                  Number of obs   =         74
GMM weight matrix:     Robust
```

	Coef.	Robust Std. Err.	z	P>|z|	[95% Conf.	Interval]
/m1	6384.682	546.142	11.69	0.000	5314.263	7455.1
/m2	2315.909	90.19414	25.68	0.000	2139.132	2492.686
/p	.2972973	.0531331	5.60	0.000	.1931583	.4014363
/a	7.065282	2.294704	3.08	0.002	2.567745	11.56282
/b1	.000972	.00036	2.70	0.007	.0002664	.0016775
/b2	-.0052477	.00178	-2.95	0.003	-.0087365	-.001759
/eta01	27.24295	1.494802	18.23	0.000	24.31319	30.1727

```
Instruments for equation 1: _cons
Instruments for equation 2: _cons
Instruments for equation 3: _cons
Instruments for equation 4: _cons
Instruments for equation 5: _cons
Instruments for equation 6: _cons
Instruments for equation 7: _cons

. mata:
------------------------------------------------- mata (type end to exit) -----
: N        = st_nobs()

: m        = st_matrix("e(b)")[,(1,2)]

: k        = length(m)
```

```
: a        = st_matrix("e(b)")[,4]
: b        = st_matrix("e(b)")[,(5,6)]'
: eta01    = st_matrix("e(b)")[,7]
: X        = st_data(., "price weight")
: D        = st_data(., "foreign")
: Y        = st_data(., "mpg")
: // compute IF for eta01
: w0       = !D :* exp(a :+ X*b)
: hm       =  D :* (X :- m)
: hb       = w0 :* (X :- m)
: heta01   = w0 :* (Y :- eta01)
: G11inv   = diag(J(k, 1, N/sum(D)))
: G41      = diag(J(k, 1, sum(w0))) / N
: G44inv   = luinv(-cross(hb, X) / N)
: G54      = -colsum(heta01 :* X) / N
: IF_eta01 = N/sum(w0) * (heta01 - (hb - hm * G11inv' * G41') * G44inv' * G54')
: // compute IF for eta11
: eta11    = mean(Y, D)
: IF_eta11 = N/sum(D) * D :* (Y :- eta11)
: // compute IF for ATT
: ATT      = eta11 - eta01
: IF_ATT   = IF_eta11 - IF_eta01
: // display results (point estimate, mean of IF, standard error)
: (ATT, eta11, eta01)', mean((IF_ATT, IF_eta11, IF_eta01))',
>     sqrt(diagonal(variance((IF_ATT, IF_eta11, IF_eta01)) / N)) * sqrt((N-1)/N)
                 1              2              3
  1    -2.470218473   -1.84290e-15     1.74221528
  2     24.77272727   -6.54131e-16    1.377102927
  3     27.24294575    1.13889e-15    1.494801663

: end
```

Entropy balancing with regression adjustment

Even though entropy balancing perfectly balances the data, regression adjustment may make sense to take account of additional covariates. Let z_i be a vector of predictors (including a constant) for the regression adjustment. In the control group, a weighted regression of Y on

Z is run to determine regression coefficients γ, that are then used to estimate the potential outcomes in the treatment group. In terms of GMM, the problem be written as

$$\mathbf{h}_i(\mathbf{X}_i;\theta) = \begin{bmatrix} h_{i1}(x_i;\mu) \\ h_{i2}(1;p) \\ h_{i3}(1;\alpha) \\ h_{i4}(x_i;\beta) \\ h_{i5}(z_i;\gamma_0) \\ h_{i6}(1;\eta_1^0) \end{bmatrix} = \begin{bmatrix} d_i(x_i' - \mu) \\ d_i - p \\ \omega_i^0 - \frac{(1-d_i)p}{1-p} \\ \omega_i^0(x_i' - \mu) \\ z_i'\omega_i^0(y_i - z_i\gamma_0) \\ d_i(z_i\gamma_0 - \eta_1^0) \end{bmatrix}$$

such that

$$\overline{\mathbf{G}}(\hat{\theta}) = \frac{1}{N}\sum_{i=1}^{N} \begin{bmatrix} d_i I_k & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & 1 & 0 & \mathbf{0} & \mathbf{0} & 0 \\ \mathbf{0} & \frac{1-d_i}{(1-\hat{p})^2} & -\hat{\omega}_i^0 & -\hat{\omega}_i^0 x_i & \mathbf{0} & 0 \\ \hat{\omega}_i^0 I_k & \mathbf{0} & h_{i4}(x_i;\hat{\beta}) & -h_{i4}(x_i;\hat{\beta})x_i & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & h_{i5}(z_i;\hat{\gamma}_0) & -h_{i5}(z_i;\hat{\gamma}_0)x_i & z_i'\hat{\omega}_i^0 z_i & \mathbf{0} \\ \mathbf{0} & 0 & 0 & \mathbf{0} & -d_i z_i & d_i \end{bmatrix}$$

The influence function for η_1^0 thus is:

$$\lambda_i(\hat{\eta}_1^0) = \frac{N}{\sum_i d_i}\left(h_{i6}(1;\hat{\eta}_1^0) - \overline{\mathbf{G}}_{65}\overline{\mathbf{G}}_{55}^{-1}\left(h_{i5}(1;\hat{\eta}_1^0) - \overline{\mathbf{G}}_{54}\overline{\mathbf{G}}_{44}^{-1}\left(h_{i4}(x_i;\hat{\beta}) - \overline{\mathbf{G}}_{41}\overline{\mathbf{G}}_{11}^{-1}h_{i1}(x_i;\hat{\mu})\right)\right)\right) \tag{27}$$

with

$$\overline{\mathbf{G}}_{65} = \frac{1}{N}\sum_{i=1}^{N} -d_i z_i \qquad \overline{\mathbf{G}}_{54} = \frac{1}{N}\sum_{i=1}^{N} -h_{i5}(1;\hat{\eta}_1^0)x_i \qquad \overline{\mathbf{G}}_{41} = \frac{1}{N}\sum_{i=1}^{N} \hat{\omega}_i^0 I_k$$

$$\overline{\mathbf{G}}_{55} = \frac{1}{N}\sum_{i=1}^{N} z_i'\hat{\omega}_i^0 z_i \qquad \overline{\mathbf{G}}_{44} = \frac{1}{N}\sum_{i=1}^{N} -h_{i4}(x_i;\hat{\beta})x_i \qquad \overline{\mathbf{G}}_{11} = \frac{1}{N}\sum_{i=1}^{N} d_i I_k$$

Example:

```
. sysuse auto, clear
(1978 Automobile Data)

. local w "(foreign==0) * exp({a} + {b1}*price + {b2}*weight)"
```

```
. local zg "({g1}*turn + {g0})"

. gmm (foreign*(price  - {m1}))           ///
>     (foreign*(weight - {m2}))           ///
>     (foreign - {p})                     ///
>     (`w' - (foreign==0)*{p}/(1-{p}))    ///
>     (`w'*(price  - {m1}))               ///
>     (`w'*(weight - {m2}))               ///
>     (turn*`w'*(mpg - `zg'))             ///
>     (     `w'*(mpg - `zg'))             ///
>     (foreign*(`zg' - {eta01}))          ///
>     , winitial(identity) nolog

Final GMM criterion Q(b) =  1.70e-30

note: model is exactly identified

GMM estimation

Number of parameters =   9
Number of moments    =   9
Initial weight matrix: Identity                  Number of obs   =         74
GMM weight matrix:     Robust
```

	Coef.	Robust Std. Err.	z	P>|z|	[95% Conf.	Interval]
/m1	6384.682	546.142	11.69	0.000	5314.263	7455.1
/m2	2315.909	90.19414	25.68	0.000	2139.132	2492.686
/p	.2972973	.0531331	5.60	0.000	.1931583	.4014363
/a	7.065282	2.294704	3.08	0.002	2.567745	11.56282
/b1	.000972	.00036	2.70	0.007	.0002664	.0016775
/b2	-.0052477	.00178	-2.95	0.003	-.0087365	-.001759
/g1	-1.064548	.3057743	-3.48	0.000	-1.663855	-.4652416
/g0	66.38138	12.28574	5.40	0.000	42.30177	90.461
/eta01	28.68669	2.497824	11.48	0.000	23.79105	33.58234

```
Instruments for equation 1: _cons
Instruments for equation 2: _cons
Instruments for equation 3: _cons
Instruments for equation 4: _cons
Instruments for equation 5: _cons
Instruments for equation 6: _cons
Instruments for equation 7: _cons
Instruments for equation 8: _cons
Instruments for equation 9: _cons

. mata:
------------------------------------------------- mata (type end to exit) -----
: N        = st_nobs()

: m        = st_matrix("e(b)")[,(1,2)]

: k        = length(m)

: a        = st_matrix("e(b)")[,4]

: b        = st_matrix("e(b)")[,(5,6)]'
```

```
: g        = st_matrix("e(b)")[,(7,8)]'
: eta01    = st_matrix("e(b)")[,9]
: X        = st_data(., "price weight")
: Z        = st_data(., "turn"), J(N, 1, 1)
: D        = st_data(., "foreign")
: Y        = st_data(., "mpg")
: // compute IF for eta01
: w0       = !D :* exp(a :+ X*b)
: hm       =  D :* (X :- m)
: hb       = w0 :* (X :- m)
: hg       = Z :* w0 :* (Y :- Z*g)
: heta01   = D :* (Z*g :- eta01)
: G11inv   = diag(J(k, 1, N/sum(D)))
: G41      = diag(J(k, 1, sum(w0))) / N
: G44inv   = luinv(-cross(hb, X) / N)
: G54      = -cross(hg, X) / N
: G55inv   = invsym(cross(Z, w0, Z) / N)
: G65      = colsum(-D :* Z) / N
: IF_eta01 = N/sum(D) * (heta01 - (hg - (hb - hm * G11inv' * G41') *
>                                 G44inv' * G54') * G55inv' * G65')
: // compute IF for eta11
: eta11    = mean(Y, D)
: IF_eta11 = N/sum(D) * D :* (Y :- eta11)
: // compute IF for ATT
: ATT      = eta11 - eta01
: IF_ATT   = IF_eta11 - IF_eta01
: // display results (point estimate, mean of IF, standard error)
: (ATT, eta11, eta01)', mean((IF_ATT, IF_eta11, IF_eta01))',
>     sqrt(diagonal(variance((IF_ATT, IF_eta11, IF_eta01)) / N)) * sqrt((N-1)/N)
                  1              2              3
  +-------------------------------------------------------+
1 |    -3.91396706    1.12002e-14    2.730144856          |
2 |    24.77272727   -6.54131e-16    1.377102927          |
3 |    28.68669433   -1.18986e-14    2.497824357          |
  +-------------------------------------------------------+

: end
```

If the predictors in the regression adjustment are identical to the entropy balancing covariates, that is, if $z_i = x_i$ for all i, then regression adjustment does not change the estimate of

η_1^0, nor does it change its standard error (this is related to the fact that entropy balancing has doubly-robust property; see Zhao and Percival, 2017). Furthermore, the standard errors from a regression adjustment estimator ignoring that the entropy-balancing weights $\hat{\omega}_i^0$ have been estimated will produce the same standard errors as the full influence function. That is, if $z_i = x_i$, the influence function can be simplified to

$$\lambda_i(\hat{\eta}_1^0) = \frac{N}{\sum_i d_i}\left(d_i(z_i\hat{\gamma}_0 - \hat{\eta}_1^0) - \overline{\mathbf{G}}_{65}\overline{\mathbf{G}}_{55}^{-1} z_i'\hat{\omega}_i^0(y_i - z_i\hat{\gamma}_0)\right)$$

4 Sampling weights, subpopulation estimation, and common support

Survey design

In applied situations, data may stem from a complex survey including sampling weights, clustering, and stratification. Clustering and stratification can be handled while computing the variance matrix from the influence functions using a standard survey estimator for the mean (or the total, depending on the scaling of the influence functions). Sampling weights, however, have to be taken into account in the computation of the components of the influence functions. Sampling weights w_i, with $w_i \geq 0$ for all observations, can be added to the GMM problem as follows:

$$E_w(\mathbf{h}_i(\mathbf{X}_i;\theta)) = \frac{1}{W}\sum_{i=1}^{N} w_i\mathbf{h}_i(\mathbf{X}_i;\theta) = \mathbf{0}$$

with

$$\overline{\mathbf{G}}(\hat{\theta}) = \frac{1}{W}\sum_{i=1}^{N} w_i \left.\frac{\partial \mathbf{h}_i(\mathbf{X}_i;\theta)}{\partial \theta'}\right|_{\theta=\hat{\theta}}$$

where $W = \sum_{i=1}^{N} w_i$ such that

$$\lambda_i(\hat{\theta}) = w_i\overline{\mathbf{G}}(\hat{\theta})^{-1}\mathbf{h}_i(\mathbf{X}_i;\hat{\theta})$$

Here is an example using logistic regression with sampling weights:

```
. webuse nhanes2f, clear
```

```
. logit highbp sex age [pw=finalwgt], nolog
Logistic regression                             Number of obs     =     10,337
                                                Wald chi2(2)      =    1188.90
                                                Prob > chi2       =     0.0000
Log pseudolikelihood =  -68668339               Pseudo R2         =     0.1086

------------------------------------------------------------------------------
             |               Robust
      highbp |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
         sex |    -.597088   .0517069   -11.55   0.000    -.6984317   -.4957443
         age |     .050979   .0015449    33.00   0.000     .0479512    .0540069
       _cons |   -1.861826   .1084412   -17.17   0.000    -2.074367   -1.649285
------------------------------------------------------------------------------

. mata:
------------------------------------------------- mata (type end to exit) -----
: N    = st_nobs()

: D    = st_data(.,"highbp")

: X    = st_data(.,"sex age"), J(N, 1, 1)

: w    = st_data(.,"finalwgt")

: W    = sum(w)

: b    = st_matrix("e(b)")'

: p    = invlogit(X * b)

: h    = X :* (D :- p)

: Ginv = invsym(cross(X, w :* (p :* (1 :- p)), X) / W)

: IF   = w :* h * Ginv'

: b, mean(IF)', sqrt(colsum(IF:^2)/(W - W/N) / W)'
                  1              2              3
  +-------------------------------------------------+
1 |  -.5970880093   -1.38760e-09    .0517069357    |
2 |   .0509790351    6.85179e-11    .0015448592    |
3 |  -1.861826094   -1.38651e-09     .108441192    |
  +-------------------------------------------------+

: end
------------------------------------------------------------------------------
```

From a practical perspective, it may be convenient set the influence function to $\lambda_i(\hat{\theta})/w_i$, that is, to leave the leading w_i away, so that the standard errors can be estimated by applying a mean estimator accounting for the survey design, including the weights (see [R] **mean**). Here is an example using logistic regression with a fully-fledged complex survey design (reusing some results from above):

```
. quietly svyset psuid [pweight=finalwgt], strata(stratid)
```

```
. svy: logit highbp sex age
(running logit on estimation sample)

Survey: Logistic regression

Number of strata   =        31                Number of obs     =       10,337
Number of PSUs     =        62                Population size   =  117,023,659
                                              Design df         =           31
                                              F(   2,     30)   =       544.16
                                              Prob > F          =       0.0000

------------------------------------------------------------------------------
             |             Linearized
      highbp |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
         sex |   -.597088   .0569272   -10.49   0.000    -.7131919   -.4809841
         age |    .050979    .001578    32.31   0.000     .0477606    .0541974
       _cons |  -1.861826   .1422783   -13.09   0.000    -2.152005   -1.571648
------------------------------------------------------------------------------

. quietly generate double IF1 = .

. quietly generate double IF2 = .

. quietly generate double IF3 = .

. mata:
------------------------------------------------- mata (type end to exit) -----
: IF = h * Ginv'  // omit weights

: st_store(.,tokens("IF1 IF2 IF3"), IF)

: end
------------------------------------------------------------------------------

. svy: mean IF1 IF2 IF3
(running mean on estimation sample)

Survey: Mean estimation

Number of strata =      31      Number of obs   =       10,337
Number of PSUs   =      62      Population size =  117,023,659
                                Design df       =           31

--------------------------------------------------------------
             |             Linearized
             |       Mean   Std. Err.     [95% Conf. Interval]
-------------+------------------------------------------------
         IF1 |  -1.23e-13   .0569272     -.1161039    .1161039
         IF2 |   6.06e-15    .001578     -.0032184    .0032184
         IF3 |  -1.23e-13   .1422783     -.2901786    .2901786
--------------------------------------------------------------
```

We see that the standard errors from `svy:mean` applied to the influence functions are identical to the standard errors from `svy:logit`.

Subpopulation estimation

When working with survey data, we may be interested in an analysis of a subpopulation and want to account for the fact that the subpopulation size is not fixed. To do so, simply multiply the moment equations by the subpopulation indicator, that is, use $\mathbf{h}_i^s(\mathbf{X}_i;\hat{\theta}) = s_i\mathbf{h}_i(\mathbf{X}_i;\hat{\theta})$ in place of $\mathbf{h}_i(\mathbf{X}_i;\hat{\theta})$ in the GMM problem. Example:

```
. webuse nhanes2f, clear
. quietly svyset psuid [pweight=finalwgt], strata(stratid)
. generate byte subpop = (sex==2) // females
. svy, subpop(subpop): logit highbp age
(running logit on estimation sample)
Survey: Logistic regression
Number of strata   =        31                Number of obs     =       10,337
Number of PSUs     =        62                Population size   =  117,023,659
                                              Subpop. no. obs   =        5,428
                                              Subpop. size      =   60,901,624
                                              Design df         =           31
                                              F(   1,     31)   =       580.65
                                              Prob > F          =       0.0000
```

highbp	Coef.	Linearized Std. Err.	t	P>|t|	[95% Conf.	Interval]
age	.0665089	.0027601	24.10	0.000	.0608797	.0721382
_cons	-3.785541	.1335468	-28.35	0.000	-4.057912	-3.51317

```
. quietly generate double IF1 = 0
. quietly generate double IF2 = 0
. mata:
                                        mata (type end to exit)
: D    = st_data(.,"highbp")
: X    = st_data(.,"age"), J(rows(D), 1, 1)
: s    = st_data(.,"subpop")
: w    = st_data(.,"finalwgt")
: W    = sum(w)
: b    = st_matrix("e(b)")'
: p    = invlogit(X * b)
: h    = s :* X :* (D :- p)
: Ginv = invsym(cross(X, w :* s :* (p :* (1 :- p)), X) / W)
: IF   = h * Ginv'
```

```
: st_store(.,tokens("IF1 IF2"), IF)
: end
```

```
. svy: mean IF1 IF2
(running mean on estimation sample)

Survey: Mean estimation

Number of strata =      31      Number of obs   =       10,337
Number of PSUs   =      62      Population size =  117,023,659
                                Design df       =           31
```

	Mean	Linearized Std. Err.	[95% Conf.	Interval]
IF1	1.26e-11	.0027601	-.0056292	.0056292
IF2	-7.26e-10	.1335468	-.2723706	.2723705

A alternative, possibly more convenient approach may be to do all computations using only the observations of the subpopulation (and set the influence function to zero for all other observations). In this case, however, one has to be careful to rescale the influence function by the relative size of the subpopulation. An easier approach is to divide the influence function by the subpopulation size and then use the `total` command instead of `mean` to compute the standard errors (see [R] **total**). Example (reusing some results from above):

```
. quietly replace IF1 = 0
. quietly replace IF2 = 0
. mata:
                                        mata (type end to exit)
: D    = st_data(.,"highbp", "subpop")
: X    = st_data(.,"age", "subpop"), J(rows(D), 1, 1)
: w    = st_data(.,"finalwgt", "subpop")
: W    = sum(w)
: p    = invlogit(X * b)
: h    = X :* (D :- p)
: Ginv = invsym(cross(X, w :* (p :* (1 :- p)), X) / W)
: IF   = (h * Ginv') / W
: st_store(.,tokens("IF1 IF2"), "subpop", IF)
: end
```

```
. svy, subpop(subpop): total IF1 IF2
```

```
(running total on estimation sample)

Survey: Total estimation

Number of strata =      31      Number of obs   =       10,337
Number of PSUs   =      62      Population size =  117,023,659
                                Subpop. no. obs =        5,428
                                Subpop. size    =   60,901,624
                                Design df       =           31
```

	Total	Linearized Std. Err.	[95% Conf.	Interval]
IF1	1.26e-11	.0027601	-.0056292	.0056292
IF2	-7.26e-10	.1335468	-.2723706	.2723705

Common support

Related to subpopulation estimation is the problem that some observations may be excluded from the treatment effect estimation because they are outside of the common support. For example, when computing an ATT in exact matching, caliper matching, or kernel matching, some observations from the treatment group may be excluded because no matching controls can be found for them. Intuitively, we may treat such a situation as a subpopulation estimation problem and use the approach above, restricting s_i to the subsample within the common support. However, note that in this way the common support indicator is treated as exogenous, which may not be appropriate. A more refined approach would treat the common support indicator as endogenous and then define the influence function accordingly. Further research will be needed to work out these details.

5 Conclusions

As has been illustrated in this paper, GMM provides a powerful and flexible framework for deriving influence functions. The framework has been successfully applied to a variety of treatment effect estimators. Some remaining issues are as follows.

- The influence functions for matching estimators presented in this paper make some simplifying assumptions. In particular, the matching configuration is assumed fixed. This may not bias the standard error estimates to badly in situations where multiple matches are used (e.g. nearest-neighbor matching with multiple neighbors, kernel matching). In 1-nearest-neighbor matching, however, the bias may be severe. It would be worthwhile to try to find an improved expression for the influence function that solves this problem.

- Propensity-score matching (PSM) has not been considered in this paper. Deriving the influence functions for PSM is more difficult than for Mahalanobis distance matching because the propensity score is estimated, which will affect the standard errors. Ignoring this fact leads to standard error estimates that are conservative.

- Some matching estimators restrict the sample to a region of common support. Subpopulation estimation may be a solution to this problem, but refined approaches could possibly be developed.

References

Abadie, Alberto and Guido W. Imbens. 2006. "Large sample properties of matching estimators for average treatment effects." *Econometrica* 74:604–620.

Angrist, Joshua D. and Jörn-Steffen Pischke. 2009. *Mostly Harmless Econometrics: An Empiricist's Companion*. Princeton: Princeton University Press.

Cameron, A. Colin and Pravin K. Trivedi. 2005. *Microeconometrics: Methods and Applications*. New York: Cambridge University Press.

Deville, Jean-Claude. 1999. "Variance estimation for complex statistics and estimators: Linearization and residual techniques." *Survey Methodology* 25:193–203.

Drukker, David M. 2014. "Using gmm to solve two-step estimation problems." The Stata Blog: Not Elsewhere Classified. Available from http://blog.stata.com/2014/12/08/using-gmm-to-solve-two-step-estimation-problems/.

Hainmueller, Jens. 2012. "Entropy Balancing for Causal Effects: A Multivariate Reweighting Method to Produce Balanced Samples in Observational Studies." *Political Analysis* 20:25–46.

Hampel, Frank R. 1974. "The Influence Curve and Its Role in Robust Estimation." *Journal of the American Statistical Association* 69:383–393.

Hampel, Frank R., Elvezio M. Ronchetti, Peter J. Rousseeuw, and Werner A. Stahel. 1986. *Robust Statistics. The Approach Based on Influence Functions*. New York: John Wiley & Sons.

jayk. 2015. "Influence functions and OLS." Cross Validated. Available from https://stats.stackexchange.com/q/139506 (version: 2015-02-26).

Imbens, Guido W. and Donald B. Rubin. 2015. *Causal inference for statistics, social, and biomedical sciences. An introduction*. New York: Cambridge University Press.

Imbens, Guido W. and Jeffrey M. Wooldridge. 2009. "Recent Developments in the Econometrics of Program Evaluation." *Journal of Economic Literature* 47:5–86.

Jann, Ben. 2017. "kmatch: Stata module for multivariate-distance and propensity-score matching, including entropy balancing, inverse probability weighting, (coarsened) exact matching, and regression adjustment." Available from https://ideas.repec.org/c/boc/bocode/s458346.html.

Jann, Ben. 2019. "Influence functions for linear regression (with an application to regression adjustment)." University of Bern Social Sciences Working Papers 32. DOI: 10.7892/boris.130362.

Kahn, Jay. 2015. "Influence Functions for Fun and Profit." Ross School of Business, University of Michigan. Available from http://j-kahn.com/files/influencefunctions.pdf (version: July 10, 2015).

Morgan, Stephen L. and Christopher Winship. 2015. *Counterfactuals and Causal Inference. Methods and Principles for Social Research. 2nd edition*. New York: Cambridge University Press.

Staudte, Robert G. and Simon J. Sheather. 1990. *Robust Estimation and Testing*. New York: John Wiley & Sons.

Zhao, Qingyuan and Daniel Percival. 2017. "Entropy Balancing is Doubly Robust." *Journal of Causal Inference* 5:2016–0010.