UNIVERSITÄT
BERN

Faculty of Business, Economics and Social Sciences

**Department of Social Sciences**

University of Bern Social Sciences Working Paper No. 39

# Entropy balancing as an estimation command

Ben Jann

August 3, 2021

http://ideas.repec.org/p/bss/wpaper/39.html
http://econpapers.repec.org/paper/bsswpaper/39.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

# Entropy balancing as an estimation command

Ben Jann
Institute of Sociology
University of Bern
ben.jann@unibe.ch

**Abstract.** Entropy balancing is a popular reweighting technique that provides an alternative to approaches such as, for example, inverse probability weighting (IPW) based on a logit or probit model. Even if the balancing weights resulting from the procedure will be of primary interest in most applications, it is noteworthy that entropy balancing can be represented as a simple regression-like model. An advantage of treating entropy balancing as a parametric model is that it clarifies how the reweighting affects statistical inference. In this article I present a new Stata command called `ebalfit` that estimates such a model including the variance-covariance matrix of the estimated coefficients. The balancing weights are then obtained as model predictions. Variance estimation is based on influence functions, which can be stored for further use, for example, to obtain consistent standard errors for statistics computed from the reweighted data.

**Keywords:** st0001, Stata, `ebalfit`, entropy balancing, reweighting, inverse probability weighting, IPW, influence function

## 1 Introduction

The goal of entropy balancing, a procedure made popular by Hainmueller (2012), is to find a vector of weights that balances the data between two subsamples with respect to specific moments (e.g. the means and variances of a given set of covariates). For example, in order to estimate an "average treatment effect on the treated" (ATET) from observational data we might want to reweight a "control group" such that the means of observed pre-treatment variables match the means of these variables in the "treatment group". Entropy balancing thus provides an alternative to other reweighting techniques commonly used in the treatment effects literature, such as inverse probability weighting (IPW) or matching (see, e.g., Imbens and Wooldridge 2009 for an overview), some of which are implemented in Stata's `teffects` command ([TE] **teffects**). An advantage of entropy balancing over classic IPW or matching is that it leads to perfect balance (if perfect balance is possible given the degree to which the common support assumption is violated); classic IPW and matching typically balance the data only approximately (unless the balancing problem is very simple). Perfect balance means that modeling the outcome (e.g. using regression adjustment) after the data have been balanced will lead to no refinements in the treatment effect estimate, implying that entropy balancing has the "doubly robust" property (also see Zhao and Percival 2017).

Entropy balancing can also be useful for other types of applications. For example, we may employ entropy balancing to construct weights for population surveys, say, by

adjusting the sample to a known population distribution or by fitting weights based on sampling frame data (see, e.g., Chapters 13 and 14 in Valliant et al. 2013 for an overview of survey weighting). Related applications would be to use entropy balancing to compensate selective attrition in a randomized controlled trial (RCT) (assuming that selection is on observables) or to generalize experimental results from a selective sample of participants to the population (assuming that treatment effect heterogeneity is conditional on observables). Furthermore, entropy balancing may be used for more peculiar purposes, such as constructing weights that make the data orthogonal.

From a statistical point of view, entropy balancing boils down to a relatively simple regression-like parametric model. A first key contribution of this paper is to show how entropy balancing can be expressed as a system of moment equations and how, based on this representation, influence functions can be derived for the parameters of the model. These influence functions can then be used to obtain a consistent estimate of the variance matrix of the entropy balancing coefficients, but also to adjust the variance estimates of statistics computed from the reweighted data to take account of the uncertainty implied by the estimation of the weights. A second key contribution is to provide a new command called `ebalfit` that implements the described methods. Advantages of `ebalfit` over existing implementations[1] are that `ebalfit` behaves like official Stata's estimation commands (similar syntax, output, and returns), that it provides standard errors and confidence intervals for the estimated coefficients, and that influence functions can be stored for further use in analyses employing the balancing weights.

This article is structured as follows. Drawing on preliminary work by Jann (2020b) and Jann (2020c), I first describe the entropy balancing model and its estimation, including the derivation of influence functions and an approach to adjust the standard errors of reweighted statistics. I then describe the syntax and options of the new command. Finally, I provide a set of examples illustrating the practical application of the new command.

# 2 The entropy balancing model

## 2.1 Two-sample balancing

In two-sample entropy balancing the goal is to reweight a sample of interest such that it has the same characteristics as some reference sample. Let $i = 1, \ldots, N$ be the index of observations across both samples. Indicator variable $S_i$ is equal to 1 for observations that belong to the primary sample, that is, to the sample that is to be reweighted, and 0 else. Furthermore, $\mathcal{S}$ is the set of indices of observations that belong to this sample (i.e. observations for which $S_i = 1$). Likewise, indicator variable $R_i$ is equal to 1 for observations that belong to the reference sample, and $\mathcal{R}$ contains its indices. Note

1. See command `ebalance` by Hainmueller and Xu (2011, 2013). Note that entropy balancing can also be performed by command `psweight` by Kranker (2019), a command that implements "covariate-balancing propensity score" (CBPS) estimation as proposed by Imai and Ratkovic (2014). Entropy balancing is formally equivalent to just-identified CBPS, leading to the same coefficients and the same balancing weights.

that $\mathcal{S}$ and $\mathcal{R}$ do not need to be disjoint nor exhaustive (for example, the two samples may overlap). Each observation has a base weight $w_i$ (e.g. a sampling weight based on the survey design) and a $k \times 1$ vector $\mathbf{x}_i$ of data. Furthermore, $W = \sum_{i=1}^{N} w_i$ is the sum of weights across the joint sample; $W_S = \sum_{i=1}^{N} S_i w_i = \sum_{i\in\mathcal{S}} w_i$ and $W_R = \sum_{i=1}^{N} R_i w_i = \sum_{i\in\mathcal{R}} w_i$ are the sums of weights in the primary sample and the reference sample, respectively.

Given the target sum of weights $\hat{\tau} = W_R = \sum_{i\in\mathcal{R}} w_i$ (i.e. the size of the reference sample) and the $k \times 1$ vector of target moments $\hat{\mu} = \frac{1}{W_R}\sum_{i\in\mathcal{R}} w_i\mathbf{x}_i$ (i.e. the means of the data in the reference sample), entropy balancing looks for an estimate of $(\beta', \alpha)'$ such that

$$\frac{1}{\hat{\tau}}\sum_{i\in\mathcal{S}} \hat{\omega}_i\mathbf{x}_i = \hat{\mu} \quad \text{and} \quad \sum_{i\in\mathcal{S}} \hat{\omega}_i = \hat{\tau} \quad \text{with} \quad \hat{\omega}_i = w_i \exp(\mathbf{x}_i'\hat{\beta} + \hat{\alpha}) \tag{1}$$

Note that $\alpha$ is just a normalizing constant ensuring that the sum of balancing weights is equal to $\hat{\tau}$. We could also set the target sum to some other (strictly positive) value, say, 1 or $W_S$. This would only affect $\alpha$, but not $\beta$.

Let $\gamma = (\mu', \tau, \beta', \alpha)'$ be the complete vector of estimates involved in the entropy balancing problem. Rearranging the above formulas for the different elements in $\gamma$ we can express the model as a system of moment equations given as

$$\frac{1}{W}\sum_{i=1}^{N} w_i\mathbf{h}_i^{\gamma}(\gamma) = \mathbf{0} \quad \text{with} \quad \mathbf{h}_i^{\gamma}(\gamma) = \begin{bmatrix} \mathbf{h}_i^{\mu}(\gamma) \\ h_i^{\tau}(\gamma) \\ \mathbf{h}_i^{\beta}(\gamma) \\ h_i^{\alpha}(\gamma) \end{bmatrix} = \begin{bmatrix} R_i(\mathbf{x}_i - \mu) \\ WR_i - \tau \\ S_i\exp(\mathbf{x}_i'\beta + \alpha)(\mathbf{x}_i - \mu) \\ S_i\left(\exp(\mathbf{x}_i'\beta + \alpha) - \frac{\tau}{W_S}\right) \end{bmatrix} \tag{2}$$

Following the approach outlined in Jann (2020b), the influence function for $\hat{\gamma}$ can thus be obtained as

$$\mathrm{IF}_i^{\hat{\gamma}} = \mathbf{G}^{-1}\mathbf{h}_i^{\gamma}(\hat{\gamma}) \quad \text{where} \quad \mathbf{G} = -\frac{1}{W}\sum_{i=1}^{N} w_i \left.\frac{\partial\mathbf{h}_i^{\gamma}(\gamma)}{\partial\gamma'}\right|_{\gamma=\hat{\gamma}} \tag{3}$$

Solving the derivatives we get

$$\mathbf{G} = \begin{bmatrix} \mathbf{G}^{\mu} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & G^{\tau} & \mathbf{0} & 0 \\ \mathbf{G}_{\mu}^{\beta} & \mathbf{0} & \mathbf{G}^{\beta} & \mathbf{G}_{\alpha}^{\beta} \\ \mathbf{0} & G_{\tau}^{\alpha} & \mathbf{G}_{\beta}^{\alpha} & G^{\alpha} \end{bmatrix} = -\frac{1}{W}\sum_{i=1}^{N} w_i \begin{bmatrix} -R_i\mathbf{I}_k & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & -1 & \mathbf{0} & 0 \\ -S_i\hat{v}_i\mathbf{I}_k & \mathbf{0} & \mathbf{h}_i^{\beta}(\hat{\gamma})\mathbf{x}_i' & \mathbf{h}_i^{\beta}(\hat{\gamma}) \\ \mathbf{0} & -S_i\frac{1}{W_S} & S_i\hat{v}_i\mathbf{x}_i' & S_i\hat{v}_i \end{bmatrix} \tag{4}$$

where $\mathbf{I}_k$ is the identity matrix of size $k$ and $\hat{v}_i = e^{\mathbf{x}_i'\hat{\beta}+\hat{\alpha}}$. We are only interested in the influence functions for $\hat{\beta}$ and $\hat{\alpha}$, so we collapse the system to

$$\begin{bmatrix} \mathrm{IF}_i^{\hat{\beta}} \\ \mathrm{IF}_i^{\hat{\alpha}} \end{bmatrix} = \begin{bmatrix} \mathbf{G}^{\beta} & \mathbf{G}_{\alpha}^{\beta} \\ \mathbf{G}_{\beta}^{\alpha} & G^{\alpha} \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{h}_i^{\beta}(\hat{\gamma}) - \mathbf{G}_{\mu}^{\beta}(\mathbf{G}^{\mu})^{-1}\mathbf{h}_i^{\mu}(\hat{\gamma}) \\ h_i^{\alpha}(\hat{\gamma}) - G_{\tau}^{\alpha}(G^{\tau})^{-1}h_i^{\tau}(\hat{\gamma}) \end{bmatrix} \tag{5}$$

Note that

$$\mathbf{G}^{\mu} = -\frac{1}{W}\sum_{i=1}^{N} w_i(-R_i\mathbf{I}_k) = \frac{W_R}{W}\mathbf{I}_k \qquad G^{\tau} = -\frac{1}{W}\sum_{i=1}^{N} w_i(-1) = 1$$
$$G^{\alpha}_{\tau} = -\frac{1}{W}\sum_{i=1}^{N} w_i(-S_i\frac{1}{W_S}) = \frac{1}{W}$$

so that (5) can be simplified to

$$\begin{bmatrix} \text{IF}_i^{\hat{\beta}} \\ \text{IF}_i^{\hat{\alpha}} \end{bmatrix} = \begin{bmatrix} \mathbf{G}^{\beta} & \mathbf{G}^{\beta}_{\alpha} \\ \mathbf{G}^{\alpha}_{\beta} & G^{\alpha} \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{h}_i^{\beta}(\hat{\gamma}) - \frac{W}{W_R}\mathbf{G}^{\beta}_{\mu}\mathbf{h}_i^{\mu}(\hat{\gamma}) \\ h_i^{\alpha}(\hat{\gamma}) - \frac{1}{W}h_i^{\tau}(\hat{\gamma}) \end{bmatrix} \tag{6}$$

Furthermore, using rules for the inversion of a block matrix, we can write

$$\begin{bmatrix} \mathbf{G}^{\beta} & \mathbf{G}^{\beta}_{\alpha} \\ \mathbf{G}^{\alpha}_{\beta} & G^{\alpha} \end{bmatrix}^{-1} = \begin{bmatrix} \mathbf{A} & -d\left(\mathbf{G}^{\beta}\right)^{-1}\mathbf{G}^{\beta}_{\alpha} \\ -\mathbf{G}^{\alpha}_{\beta}\mathbf{A}/G^{\alpha} & d \end{bmatrix} \quad \text{with} \quad \begin{array}{l} \mathbf{A} = \left(\mathbf{G}^{\beta} - \mathbf{G}^{\beta}_{\alpha}\mathbf{G}^{\alpha}_{\beta}/G^{\alpha}\right)^{-1} \\ d = \frac{1}{G^{\alpha} - \mathbf{G}^{\alpha}_{\beta}(\mathbf{G}^{\beta})^{-1}\mathbf{G}^{\beta}_{\alpha}} \end{array} \tag{7}$$

such that the influence functions are given as

$$\text{IF}_i^{\hat{\beta}} = \mathbf{A}\left(\mathbf{h}_i^{\beta}(\hat{\gamma}) - \tfrac{W}{W_R}\mathbf{G}^{\beta}_{\mu}\mathbf{h}_i^{\mu}(\hat{\gamma})\right) - d\left(\mathbf{G}^{\beta}\right)^{-1}\mathbf{G}^{\beta}_{\alpha}\left(h_i^{\alpha}(\hat{\gamma}) - \frac{1}{W}h_i^{\tau}(\hat{\gamma})\right) \tag{8}$$

$$\text{IF}_i^{\hat{\alpha}} = d\left(h_i^{\alpha}(\hat{\gamma}) - \frac{1}{W}h_i^{\tau}(\hat{\gamma})\right) - \frac{\mathbf{G}^{\alpha}_{\beta}\mathbf{A}}{G^{\alpha}}\left(\mathbf{h}_i^{\beta}(\hat{\gamma}) - \tfrac{W}{W_R}\mathbf{G}^{\beta}_{\mu}h_i^{\mu}(\hat{\gamma})\right) \tag{9}$$

If balance is achieved, then

$$\mathbf{G}^{\beta}_{\mu} = -\frac{1}{W}\sum_{i=1}^{N} w_i(-S_i\hat{v}_i\mathbf{I}_k) = \frac{\hat{\tau}}{W}\mathbf{I}_k \qquad \mathbf{G}^{\beta}_{\alpha} = -\frac{1}{W}\sum_{i=1}^{N} w_i\mathbf{h}_i^{\beta}(\hat{\gamma}) = \mathbf{0}$$
$$G^{\alpha} = -\frac{1}{W}\sum_{i=1}^{N} w_iS_i\hat{v}_i = -\frac{\hat{\tau}}{W}$$

such that (8) and (9) simplify to

$$\text{IF}_i^{\hat{\beta}} = (\mathbf{G}^{\beta})^{-1}\left(\mathbf{h}_i^{\beta}(\hat{\gamma}) - \frac{\hat{\tau}}{W_R}\mathbf{h}_i^{\mu}(\hat{\gamma})\right) \tag{10}$$

$$\text{IF}_i^{\hat{\alpha}} = \frac{W}{-\hat{\tau}}\left(h_i^{\alpha}(\hat{\gamma}) - \frac{1}{W}h_i^{\tau}(\hat{\gamma}) - \mathbf{G}^{\alpha}_{\beta}\text{IF}_i^{\hat{\beta}}\right) \tag{11}$$

In the current setup, note that $\hat{\tau}/W_R = 1$, but we may wish to normalize the weights using some other value for $\hat{\tau}$, in which case $\hat{\tau}/W_R$ would no longer be equal to 1. For example, we may set $\hat{\tau}$ to the sum of base weights in the primary sample, that is, $\hat{\tau} = W_S = \sum_{i\in\mathcal{S}} w_i$. In this case, use $h_i^{\tau}(\gamma) = WS_i - \tau$ in (9) or (11) instead of $h_i^{\tau}(\gamma) = WR_i - \tau$. Alternatively, we may want to set $\tau$ to some fixed value, such as

$\tau = 1$. In this case, $h_i^\tau(\gamma) = 0$. Yet, an advantage of using $\hat\tau = W_R$ is that, in this case, $\hat p_i = \exp(\mathbf{x}_i'\hat\beta + \hat\alpha)/(1 + \exp(\mathbf{x}_i'\hat\beta + \hat\alpha))$ can be interpreted as a propensity score, that is, as an estimate of the conditional probability of belonging to $\mathcal{R}$ rather than $\mathcal{S}$ given $\mathbf{x}_i$.

In general, it seems justifiable to assume $\tau$ as fixed even when it is set to sample quantities such as $W_R$ or $W_S$. First, the moment condition for $\tau$ will only affect the influence function of $\hat\alpha$, which is typically only of minor interest (for example, the influence function of $\hat\alpha$ is typically not needed when correcting the standard errors of statistics computed from the reweighted data). Second, also for the influence function of $\hat\alpha$ the bias introduced by assuming $\tau$ as fixed will typically be small. This is why command `ebalfit` discussed below will treat $\tau$ as fixed when computing influence functions and standard errors.

## 2.2 One-sample balancing

In one-sample entropy balancing, the data is adjusted to given values from an external source (e.g. known population averages). To obtain the influence functions for this situation, replace $\hat\mu$ by fixed vector $\mu$ and replace $\hat\tau$ by fixed value $\tau$ (e.g. the population size) and let $\gamma = (\beta', \alpha)'$. Compared to the two-sample case, all components related to the estimation of $\mu$ and $\tau$ drop out of the system. Hence, we get

$$\mathrm{IF}_i^{\hat\beta} = \mathbf{A}\mathbf{h}_i^\beta(\hat\gamma) - d\left(\mathbf{G}^\beta\right)^{-1}\mathbf{G}_\alpha^\beta\, h_i^\alpha(\hat\gamma) \tag{12}$$

$$\mathrm{IF}_i^{\hat\alpha} = d\, h_i^\alpha(\hat\gamma) - \frac{\mathbf{G}_\beta^\alpha\mathbf{A}}{G^\alpha}\mathbf{h}_i^\beta(\hat\gamma) \tag{13}$$

and, if balance is achieved,

$$\mathrm{IF}_i^{\hat\beta} = (\mathbf{G}^\beta)^{-1}\mathbf{h}_i^\beta(\hat\gamma) \tag{14}$$

$$\mathrm{IF}_i^{\hat\alpha} = \frac{W}{-\tau}\left(h_i^\alpha(\hat\gamma) - \mathbf{G}_\beta^\alpha\mathrm{IF}_i^{\hat\beta}\right) \tag{15}$$

## 2.3 Estimation

We could use `gmm` ([R] **gmm**) to estimate the entropy balancing coefficients based on the moment equations provided above. However, given that $\alpha$ is simply a normalization constant, it may be more convenient to first run an optimization algorithm to fit $\hat\beta$ and then determine $\hat\alpha$ as

$$\hat\alpha = \ln(\tau) - \ln\left(\sum_{i\in\mathcal{S}} w_i \exp(\mathbf{x}_i'\hat\beta)\right) \tag{16}$$

This ensures that the sum of balancing weights will always equal the target sum of weights. Furthermore, in the two-sample case, the complexity of the estimation can be reduced by computing the target means $\mu$ and the target sum of weights $\tau$ upfront instead of including them in a joint optimization problem.

To obtain an estimate for $\beta$, we can run a standard Newton-Raphson algorithm that minimizes

$$L^{\beta} = \ln\left(\sum_{i\in\mathcal{S}} w_i \exp((\mathbf{x}_i - \mu)'\beta)\right) = \ln\left(\sum_{i\in\mathcal{S}} \tilde{\omega}_i\right) \quad \text{where} \quad \tilde{\omega}_i = w_i \exp((\mathbf{x}_i - \mu)'\beta) \tag{17}$$

with respect to $\beta$ (also see Hainmueller 2012). The vector of first derivatives of $L^{\beta}$ (the gradient vector) and the matrix of second derivatives (the Hessian), which are required by the Newton-Raphson procedure, are given as

$$g = \frac{1}{\sum_{i\in\mathcal{S}} \tilde{\omega}_i} \sum_{i\in\mathcal{S}} \tilde{\omega}_i(\mathbf{x}_i - \mu) \quad \text{and} \quad H = \frac{1}{\sum_{i\in\mathcal{S}} \tilde{\omega}_i} \sum_{i\in\mathcal{S}} \tilde{\omega}_i(\mathbf{x}_i - \mu)(\mathbf{x}_i - \mu)' \tag{18}$$

In practice, to avoid numerical overflow, we may want to change the definition of $\tilde{\omega}$ to

$$\tilde{\omega}_i = w_i \exp((\mathbf{x}_i - \mu)'\beta - c) \quad \text{where} \quad c = \max((\mathbf{x}_i - \mu)'\beta) \tag{19}$$

and redefine $L^{\beta}$ as

$$L^{\beta} = \ln\left(\sum_{i\in\mathcal{S}} \tilde{\omega}_i\right) + c \tag{20}$$

Furthermore, instead of using $L^{\beta}$, one may also determine convergence based on a loss criterion that is directly defined in terms of achieved balance, while still employing the gradient vector and Hessian given in (18) for updating $\beta$. For example, we could use the maximum absolute difference

$$L^{\text{absdif}} = \max\left(\text{abs}\left(g\right)\right) \tag{21}$$

the maximum relative difference

$$L^{\text{reldif}} = \max\left(\frac{\text{abs}\left(g\right)}{\text{abs}\left(\mu\right) + 1}\right) \tag{22}$$

or the norm

$$L^{\text{norm}} = \|g\| = \sqrt{g'g} \tag{23}$$

where $g$ is the gradient vector as defined above. As can easily be seen from its definition, the gradient vector

$$g = \frac{1}{\sum_{i\in\mathcal{S}} \tilde{\omega}_i} \sum_{i\in\mathcal{S}} \tilde{\omega}_i(\mathbf{x}_i - \mu) = \left(\frac{1}{\sum_{i\in\mathcal{S}} \tilde{\omega}_i} \sum_{i\in\mathcal{S}} \tilde{\omega}_i\mathbf{x}_i\right) - \mu \tag{24}$$

is equal to the difference between the means of the reweighted data and the target values $\mu$, given the current current values of $\beta$. That is, $g$ quantifies for each variable how well the balancing has been achieved up to that point in the algorithm.

Practical experience indicates that using one of these balancing loss criteria instead of $L^{\beta}$ makes the algorithm more robust in situations where perfect balance is not possible. However, as the optimization criterion is no longer fully consistent with the used

gradient and Hessian, the algorithm profits from some standardization of the data (so that the different variables have similar scales). For example, we may obtain the standard deviations

$$\sigma_S = \sqrt{\frac{1}{W_S}\sum_{i\in\mathcal{S}} w_i(\mathbf{x}_i - \overline{\mathbf{x}}_S)^2} \quad \text{with} \quad \overline{\mathbf{x}}_S = \frac{1}{W_S}\sum_{i\in\mathcal{S}} w_i\mathbf{x}_i \tag{25}$$

from the primary sample and then use $\mathbf{x}_i/\sigma_S$ and $\mu/\sigma_S$ instead of $\mathbf{x}_i$ and $\mu$ in equations (17) to (24). Before computing $\hat{\alpha}$ in (16), back-transform the resulting estimate for $\beta$ by dividing it by $\sigma_S$.

Furthermore, as usual, collinear terms have to be excluded from estimation. These terms, however, are relevant for the evaluation of final quality of the achieved balancing (collinear terms may remain unbalanced). My suggestion thus is to use $\mathbf{x}_i^{\text{nc}}$, a variant of $\mathbf{x}_i$ without elements that are collinear in $\mathcal{S}$, for estimation of $\beta$ (with elements corresponding to collinear terms set to 0) and then evaluate the final fit based on the complete data by applying one of the above loss functions to

$$\hat{g} = \frac{1}{\sum_{i\in\mathcal{S}}\hat{\omega}_i}\sum_{i\in\mathcal{S}}\hat{\omega}_i\mathbf{x}_i - \mu \quad \text{with} \quad \hat{\omega}_i = w_i\exp(\mathbf{x}_i'\hat{\beta} + \hat{\alpha}) \tag{26}$$

The variance-covariance matrix of $\hat{\gamma} = (\hat{\beta}', \hat{\alpha})'$ can be estimated by taking the total of the squared influence functions divided by the sum of weights. To be precise, if the base weights are frequency weights (or if there are no base weights, i.e. $w_i = 1$ for all observations and $W = N$), then

$$\widehat{V}(\hat{\gamma}) = \frac{W}{W-k-1}\sum_{i=1}^{N} w_i\lambda_i\lambda_i' \quad \text{with} \quad \lambda_i = \frac{1}{W}\begin{bmatrix}\text{IF}_i^{\hat{\beta}} \\ \text{IF}_i^{\hat{\alpha}}\end{bmatrix} \tag{27}$$

If the base weights are probability (sampling) weights, then

$$\widehat{V}(\hat{\gamma}) = \frac{N}{N-k-1}\sum_{i=1}^{N} w_i^2\lambda_i\lambda_i' \quad \text{with} \quad \lambda_i = \frac{1}{W}\begin{bmatrix}\text{IF}_i^{\hat{\beta}} \\ \text{IF}_i^{\hat{\alpha}}\end{bmatrix} \tag{28}$$

Complex survey design such as clustering or stratification can be taken into account by appropriately modifying the aggregation. In practice, variance estimates can be obtained by applying command [R] **total** to $\lambda_i$, possibly including the [SVY] **svy** prefix.

## 2.4 Balancing of higher-order moments and covariances

In the exposition above I only considered balancing of first moments (i.e. the means), but entropy balancing can easily be extended to higher moments such as the variance or the skewness or even to covariances. The balancing constraints for higher moments and covariances are equivalent to first-moment balancing constraints for specific transformations of the variables. Rather than extending the above exposition to cover higher

moments, we may thus simply change the definition of the data. To balance the variance of a variable $X$ in addition to its mean, add $x_i^2$ to the data. To also balance the skewness, add $x_i^3$. To balance the covariance between two variables $X_1$ and $X_2$, include the product $x_{i1}x_{i2}$ in the data.

## 2.5 Correcting standard errors of reweighted estimators

Define $\hat{\omega}_i = w_i\hat{\upsilon}_i$ with

$$\hat{\upsilon}_i = \begin{cases} \exp(\mathbf{x}_i'\hat{\beta} + \hat{\alpha}) & \text{if } S_i = 1 \\ 1 & \text{else} \end{cases} \tag{29}$$

That is, for observations within the reweighted sample, $\hat{\omega}_i$ is equal to the balancing weight, for all other observations, $\hat{\omega}_i$ is equal to the base weight. Most estimators can be expressed as a system of moment equations

$$\frac{1}{\sum_{i=1}^N \hat{\omega}_i} \sum_{i=1}^N \hat{\omega}_i \mathbf{h}_i^\theta(\theta) = \mathbf{0} \tag{30}$$

such that $\hat{\omega}_i$ does not appear in $\mathbf{h}_i^\theta(\theta)$. For such estimators, the necessary correction to take account of the uncertainty imposed by the estimation of the balancing weights has a very simple form. Re-expressing the system as

$$\frac{1}{W}\sum_{i=1}^N w_i \left(\frac{\hat{\upsilon}_i}{c}\mathbf{h}_i^\theta(\theta)\right) = \mathbf{0} \quad \text{with} \quad c = \frac{1}{W}\sum_{i=1}^N \hat{\omega}_i \tag{31}$$

we see that we can obtain the adjusted influence function as

$$\mathrm{IF}_i^{\hat{\theta}} = \frac{\hat{\upsilon}_i}{c}\widetilde{\mathrm{IF}}_i^{\hat{\theta}} - \mathbf{G}_\beta^{\widetilde{\mathrm{IF}}}\mathrm{IF}^{\hat{\beta}} - \mathbf{G}_\alpha^{\widetilde{\mathrm{IF}}}\mathrm{IF}^{\hat{\alpha}} \tag{32}$$

with

$$\mathbf{G}_\beta^{\widetilde{\mathrm{IF}}} = -\frac{1}{W}\sum_{i=1}^N w_i S_i \frac{\hat{\upsilon}_i}{c}\widetilde{\mathrm{IF}}_i^{\hat{\theta}}\mathbf{x}_i' \qquad \mathbf{G}_\alpha^{\widetilde{\mathrm{IF}}} = -\frac{1}{W}\sum_{i=1}^N w_i S_i \frac{\hat{\upsilon}_i}{c}\widetilde{\mathrm{IF}}_i^{\hat{\theta}} \tag{33}$$

where $\widetilde{\mathrm{IF}}_i^{\hat{\theta}}$ is the influence function of $\hat{\theta}$ assuming the weights $\hat{\omega}_i$ as fixed. Since $\mathbf{G}_\alpha^{\widetilde{\mathrm{IF}}} = \mathbf{0}$ by definition, the corrected influence function simplifies to

$$\mathrm{IF}_i^{\hat{\theta}} = \frac{\hat{\upsilon}_i}{c}\widetilde{\mathrm{IF}}_i^{\hat{\theta}} - \mathbf{G}_\beta^{\widetilde{\mathrm{IF}}}\mathrm{IF}^{\hat{\beta}} \tag{34}$$

To summarize, we can fist compute the influence function for $\hat{\theta}$ in the usual way, as if balancing weights were fixed, and then adjust the influence function using equation (34). Naturally, we need a way to obtain the (unadjusted) influence function of our estimator in the first place, but in many cases this is not very diffcult (for example, see Jann 2020b for practical instruction on how to obtain influence functions for maximum-likelihood models given the results returned by Stata).[2]

2. In the above derivation I assumed $c$, which depends on the relative size of the reweighted group (i.e. the sum of balancing weights) with respect to the size (sum of base weights) of the rest of

# 3 Stata implementation

Command `ebalfit`, available from the SSC Archive, implements the methods described above. To install the command on your system, type

```
. ssc install ebalfit
```

The heavy lifting is done by Mata function `mm_ebalance()` that is provided as part of the `moremata` library (Jann 2005), also available from the SSC Archive. To be able to run `ebalfit`, the latest update of `moremata` is required. To install `moremata`, type

```
. ssc install moremata, replace
```

The exposition below focuses on Stata command `ebalfit` and does not provide details on Mata function `mm_ebalance()`. Users interested in applying `mm_ebalance()` directly can type `help mata mm_ebalfit()` after installation to view its documentation.

## 3.1 Syntax

Syntax 1: adjust a subsample to values from another subsample (two-sample balancing)

`ebalfit` *varlist* [ *if* ] [ *in* ] [ *weight* ], `by(`*varname*`)` [ *options* ]

Syntax 2: adjust a sample to population values (one-sample balancing)

`ebalfit` *varlist* [ *if* ] [ *in* ] [ *weight* ], `population(`[ *size*`:` ]*numlist*`)` [ *options* ]

Replay results

`ebalfit` [ `,` *reporting_options* ]

where *reporting_options* are as described under "Reporting" in Section 3.2.

Generate predictions after estimation

`predict` [ *type* ] *newvar* [ *if* ] [ *in* ] [ `,` *predict_options* ]

where *predict_options* are

| | |
|---|---|
| `w` | generate balancing weights (the default) |
| `u` | generate raw balancing weights (i.e. without base weights) |
| `pr` | generate propensity scores |
| `pscore` | synonym for `pr` |
| `xb` | generate linear predictions |

the data, to be fixed. This is valid as long as the statistic conditions on $S_i$ such that the sum of balancing weights does not matter or if $\tau = W_S$ such that $c$ is always equal to 1. In other cases the true correction would be more complicated, but the bias introduced by assuming $c$ as fixed should be negligible in most situations.

Generate influence functions after estimation

`predict` [ *type* ] { *stub*`*` | *newvarlist* } [ *if* ] [ *in* ], `ifs` [ *IF_options* ]

where *IF_options* are

| | |
|---|---|
| `nocons` | skip influence function for $\alpha$; only relevant with *stub*`*` |
| `noalpha` | synonym for `nocons` |

In both syntax 1 and syntax 2, *varlist* may contain factor variables (see [U] **11.4.3 Factor variables**). `fweight`s, `pweight`s, and `iweight`s are allowed (see [U] **11.1.6 weight**).

## 3.2 Options

### Main

`by(`*groupvar*`)` is required in syntax 1 and identifies the subsamples. *groupvar* must be integer and nonnegative and must identify exactly two groups. By default, the lower value identifies the subsample to be reweighted and the higher value identifies the reference subsample. Also see option `swap`.

`swap` swaps the subsamples (only allowed in syntax 1). By default, the lower value of *groupvar* identifies the subsample to be reweighted. Specify `swap` to use the higher value of *groupvar* as the subsample to be reweighted.

`pooled` uses the pooled sample across both groups as the reference sample (only allowed in syntax 1). If `pooled` is specified, the selected subsample will be reweighted with respect to the overall sample (rather than with respect to the other subsample).

`population(`*spec*`)` is required in syntax 2. Use this option to specify the size of the population as well the population averages to which the sample should be reweighted. The syntax of *spec* is

[ *popsize*`:` ] *numlist*

where *popsize* is the size of the population and *numlist* provides the population averages of the variables. *numlist* must contain one value for each variable. If *popsize* is omitted, it will be set to the sum of weights in the sample.

`tau(`*spec*`)` specifies a custom target sum of weights for the balancing weights within the reweighted sample. *spec* may either be real number ($\# > 0$) or one of `Wref` (sum of base weights in the reference sample), `W` (sum of base weights in the reweighted sample), `Nref` (number of rows the reference sample), or `N` (number of rows the reweighted sample). The default is `Wref`.

`targets(`*options*`)` specifies the types of moments to be balanced. *options* are:

| | |
|---|---|
| mean | balance means (the default) |
| variance | balance variances (implies mean) |
| skeweness | balance skewnesses (implies mean and variance) |
| covariance | balance covariances (implies mean) |

By default, only the means of the specified variables will be balanced. If you type, for example, `targets(variance)`, then the variances of the variables will be balanced in addition to the means. Balancing of higher moments and covariances is implemented by adding extra terms to *varlist* before running the balancing algorithm. For example, `variance` will add `c.`*varname*`#c.`*varname* for each continuous variable in *varlist* (skipping omitted terms). Likewise, `covariance` will add `c.`*varname1*`#c.`*varname2* for each combination of continuous variables. Factor variables will be ignored by `variance` and `skewness`, but `covariance` will consider them and add appropriate interaction terms such as `1.`*fvvar*`#c.`*varname* (skipping base levels).

If option `targets()` is specified, interaction terms such as `i.`*fvvar*`#c.`*varname* are not allowed in *varlist*. However, interactions are allowed if option `targets()` is omitted. For example, you could type

```
c.hours##c.tenure i.south i.south#c.tenure
```

to balance the means of `hours` and `tenure`, the covariance between `hours` and `tenure`, the proportions of the levels of `south`, as well as the averages of `tenure` within levels of `south` (see [U] **11.4.3 Factor variables** for details on notation). That is, you can use custom interactions as an alternative to option `targets()` if you want to have more control over the exact configuration of moments to be balanced.

### Reporting

`level(#)` specifies the confidence level, as a percentage, for confidence intervals. The default is `level(95)` or as set by `set level` (see [R] **level**).

`noheader` suppresses the display of the table header.

`notable` suppresses the display of the table of results.

*display_options* are standard reporting options such as `eform`, `cformat()`, or `coeflegend`; see [R] ***eform_option*** and the Reporting options in [R] **Estimation options**.

`baltab[ (`*options*`) ]` displays a balancing table in addition to the table of coefficients. The balancing table contains for each term the raw mean and the reweighted mean, the target value for the reweighted mean, as well as the absolute difference and the "relative" difference (see function `reldif()` in [FN] **Mathematical functions**) between the reweighted mean and the target value. *options* are as described in [P] **_matrix_table**.

### VCE/SE

`vce(`*vcetype*`)` determines how standard errors are computed. *vcetype* may be:

`robust`
`cluster` *clustvar*
`none`

`vce(robust)`, the default, computes standard errors based on influence functions. Likewise, `vce(cluster` *clustvar*`)` computes standard errors based on influence function allowing for intragroup correlation, where *clustvar* specifies to which group each observation belongs. `vce(none)` omits the computation of standard errors.

`cluster(`*clustvar*`)` can be used as a synonym for `vce(cluster` *clustvar*`)`.

`nose` omits the computation of standard errors. Use this option to save computer time. `nose` is a synonym for `vce(none)`.

### Generate

`generate(`*newvar*`)` stores the balancing weights in *newvar*. Alternatively, use command `predict` to generate the balancing weights after estimation.

In syntax 1, weights will be filled in for both the reweighted subsample and the reference subsample, using a copy of the base weights for the latter (or 1 if there are no base weights).

`ifgenerate(`*names*`)` stores the influence functions of the coefficients. *names* is either a list of (new) variable names or *stub*`*` to create names *stub*`1`, *stub*`2`, etc. Alternatively, use command `predict` with option `ifs` to generate the influence functions after estimation. In any case, the influence functions will be scaled in a way such that command [R] **total** can be used to estimate the variance-covariance matrix (that is, compared to the expressions provided above, the stored influence functions will be divided by the sum of weights in the overall sample).

`nodescribe` suppresses the list of generated variables that is displayed in the output by default when `generate()` or `ifgenerate()` is specified.

`replace` allows replacing existing variables.

### Optimization

`btolerance(#)` sets the balancing tolerance. Balance is achieved if the balancing loss is smaller than the balancing tolerance. The default is `btolerance(1e-6)`.

`ltype(`*ltype*`)` sets the type of loss function to be used. *ltype* can be `reldif` (maximum relative difference), `absdif` (maximum absolute difference), or `norm` (norm of differences). The default is `reldif` .

`alteval` uses optimization criterion $L^{\beta}$ from equation (17) instead of the balancing loss from `ltype()`. Balancing loss will only be used to evaluate the final fit.

`iterate(#)` specifies the maximum number of iterations. Error will be returned if convergence is not reached within the specified maximum number of iterations. The

default is as set by `set maxiter` ([R] ***set_iter***).

`ptolerance(#)` specifies the convergence tolerance for the coefficient vector. Convergence is reached if `ptolerance()` or `vtolerance()` is satisfied. See [M–5] **optimize()** for details. The default is `ptolerance(1e-6)`.

`vtolerance(#)` specifies the convergence tolerance for the balancing loss. Convergence is reached if `ptolerance()` or `vtolerance()` is satisfied. See [M–5] **optimize()** for details. The default is `vtolerance(1e-7)` or, if `alteval` has been specified, `vtolerance(1e-10)`.

`difficult` uses a different stepping algorithm in nonconcave regions. See the singular H methods in [M–5] **optimize()** and the description of the `difficult` option in [R] **Maximize**.

`nostd` omits standardization of the data during estimation. Specifying `nostd` is not recommended (unless `alteval` is also specified).

`nolog` suppresses the display of progress information.

`relax` causes `ebalfit` to proceed even if convergence or balance is not achieved. `ebalfit` uses formulas assuming balance when computing influence functions and standard errors. The stored influence functions and reported standard errors will be invalid if balance has not been achieved.

`nowarn` suppresses any "convergence not achieved" or "balance not achieved" messages. This is only relevant if option `relax` has been specified.

### 3.3 Stored results

`ebalfit` stores its results in `e()` similar to any other estimation command (see [R] **Stored results**). See `help ebalfit` for a complete list of saved results.

## 4 Examples

### 4.1 Balancing two samples

Consider the data from LaLonde (1986), provided by Dehejia and Wahba (1999) at http://users.nber.org/ rdehejia/nswdata.html. The following code combines a subset of the treatment group from the NSW training program with one of the PSID comparison groups.

```
. use http://users.nber.org/~rdehejia/data/nsw_dw.dta, clear
. keep if treat==1
(260 observations deleted)
. append using http://users.nber.org/~rdehejia/data/psid_controls2.dta
. drop data_id
```

For purpose of exposition, I additionally generate some sampling weights (normalized such that the group sizes are preserved). I also set the default storage type for new variables to `double` so that some of the results below will have less roundoff error.

```
. set type double
. set seed 32387939
. generate w0 = runiform()
. summarize w0 if treat==0, meanonly
. quietly replace w0 = w0 * r(N) / r(sum) if treat==0
. summarize w0 if treat==1, meanonly
. quietly replace w0 = w0 * r(N) / r(sum) if treat==1
```

The focus of the LaLonde data lies on the comparison of `re78` (real earnings in 1978 after the program intervention) between the (experimental) treatment group and the (non-experimental) control group. The comparison is not straight forward as there are substantial differences between the two groups in terms of pre-treatment characteristics. Members of the treatment group are younger, more often black, less often married, more often without college degree, and have lower pre-treatment earnings than members of the control group:

```
. table () (treat) [pw=w0], stat(mean age-re75) nototal
```

| | treat | |
|---|---|---|
| | 0 | 1 |
| age | 37.39198 | 26.02386 |
| education | 10.50566 | 10.52379 |
| black | .4359494 | .8572367 |
| hispanic | .0722888 | .0567015 |
| married | .7481573 | .1944541 |
| nodegree | .5295137 | .6608205 |
| re74 | 10401.03 | 2230.392 |
| re75 | 7230.567 | 1682.258 |

Various techniques such as matching or inverse probability weighting (IPW) have been proposed in the literature to address the problem of making the groups comparable such that the average effect of program participation (the ATET) can be estimated consistently. Inverse probability weights, for example, could be obtained as follows:

```
. logit treat age-re75 [pw=w0], nolog
Logistic regression                                     Number of obs =    438
                                                        Wald chi2(8)  =  93.08
                                                        Prob > chi2   = 0.0000
Log pseudolikelihood = -159.20379                       Pseudo R2     = 0.4663
```

| treat | Coefficient | Robust std. err. | z | P>\|z\| | [95% conf. | interval] |
|---|---|---|---|---|---|---|
| age | -.092245 | .0158144 | -5.83 | 0.000 | -.1232408 | -.0612493 |
| education | .0642543 | .0870386 | 0.74 | 0.460 | -.1063382 | .2348467 |
| black | 1.932721 | .4336358 | 4.46 | 0.000 | 1.08281 | 2.782631 |

```
    hispanic |   1.671454    .543556     3.08   0.002     .6061038    2.736804
     married |  -1.290829   .3192307    -4.04   0.000     -1.91651    -.665148
    nodegree |   .2890979   .4642715     0.62   0.533    -.6208574    1.199053
        re74 |  -.0000947   .0000395    -2.40   0.017    -.0001721   -.0000172
        re75 |  -.0000944    .000078    -1.21   0.226    -.0002472    .0000584
       _cons |   1.649338   1.529141     1.08   0.281    -1.347723    4.646399
------------------------------------------------------------------------------
```

```
. predict pscore if treat==0, pr
(185 missing values generated)
. generate ipw = w0 * cond(treat==0, pscore/(1-pscore), 1)
. drop pscore
. table () (treat) [pw=ipw], stat(mean age-re75 re78) nototal
```

| | treat | |
|---|---|---|
| | 0 | 1 |
| age | 25.21257 | 26.02386 |
| education | 10.70952 | 10.52379 |
| black | .8905226 | .8572367 |
| hispanic | .0234441 | .0567015 |
| married | .1699626 | .1944541 |
| nodegree | .5972221 | .6608205 |
| re74 | 2814.793 | 2230.392 |
| re75 | 2433.187 | 1682.258 |
| re78 | 5088.788 | 6004.657 |

This worked quite well and much of the group differences disappeared, but there are still some non-negligible discrepancies, especially with respect to pre-treatment earnings. We can now try to improve the reweighting using entropy balancing:

```
. ebalfit age-re75 [pw=w0], by(treat)
Iteration 0:   balancing loss =  .88095577
Iteration 1:   balancing loss =  .20574871
Iteration 2:   balancing loss =  .11227971
Iteration 3:   balancing loss =  .01088361
Iteration 4:   balancing loss =  .00056568
Iteration 5:   balancing loss =  1.833e-06
Iteration 6:   balancing loss =  1.884e-11
Iteration 7:   balancing loss =  9.108e-17
Final fit:     balancing loss =  2.038e-16
Entropy balancing                                 Number of obs     =        438
                                                  Wald chi2(8)      =      60.71
                                                  Prob > chi2       =     0.0000
                                                  Balancing loss    =  2.038e-16
                                                  Loss type         =     reldif
Sample    = 0.treat (253 obs)                     CV of weights     =  3.3617903
Reference = 1.treat (185 obs)                     DEFF of weights   =  12.301634
------------------------------------------------------------------------------
             |               Robust
             | Coefficient  std. err.      z    P>|z|     [95% conf. interval]
-------------+----------------------------------------------------------------
         age |  -.0962814   .0299907    -3.21   0.001     -.155062   -.0375008
   education |   .0894269   .1486163     0.60   0.547    -.2018557    .3807096
       black |   1.640353   .5943787     2.76   0.006     .4753918    2.805314
    hispanic |   2.386918   .7496026     3.18   0.001      .917724    3.856112
```

```
    married |  -1.079562   .4866648    -2.22   0.027    -2.033407   -.1257165
  nodegree  |   .8138536   .7268795     1.12   0.263    -.6108041    2.238511
       re74 |   -.000121   .0000509    -2.38   0.017    -.0002208   -.0000212
       re75 |   -.000174   .0000818    -2.13   0.033    -.0003343   -.0000136
      _cons |   1.585729    2.51717     0.63   0.529    -3.347834    6.519292
------------------------------------------------------------------------------
```

Option `by()` identifies the groups to be compared; the specified variable must be dichotomous (e.g. 0 and 1). By default, `ebalfit` takes the group with the lower value as the group to be reweighted and takes the other group as the reference group. Specify option `swap` to switch the groups.

The coefficients displayed by `ebalfit` are similar to the coefficients of the `logit` model above. In fact, the coefficients do have a similar interpretation: a positive effect means that people with high values on the respective variable tend to be overrepresented in the reference group (and vice versa).

The output contains some more information that is relevant. For example, the "balancing loss" is a measure of how well `ebalfit` managed to balance the data. In the current situation, perfect balancing could be achieved as the balancing loss is essentially zero.[3] Furthermore, some information on the distribution of the weights is provided. CV is the coefficient of variation of the weights, defined as

$$\text{CV} = \frac{\sqrt{\frac{1}{N_S}\sum_{i\in\mathcal{S}}(\hat{\omega}_i - \overline{\omega}_S)^2}}{\overline{\omega}_S} \quad \text{with} \quad \overline{\omega}_S = \frac{1}{N_S}\sum_{i\in\mathcal{S}}\hat{\omega}_i$$

where summation is across the reweighted group ($N_S$ is the number of observations in the reweighted group); DEFF is the "design effect" of the weights based on Kish's formula for the effective sample size (Kish 1965), that is

$$\text{DEFF} = \frac{N_S \sum_{i\in\mathcal{S}} \hat{\omega}_i^2}{\left(\sum_{i\in\mathcal{S}} \hat{\omega}_i\right)^2}$$

Both statistics indicate that there is large variation in the weights. Apparently, the two groups are very different and balancing them is an ambitious exercise.

As mentioned, however, despite the difficulty of the problem, the output by `ebalfit` tells us that perfect balance has been achieved. We can confirm that this is true by replaying results with option `baltable` to displaying the balancing table that is provided by `ebalfit` (but is suppressed in the output by default):

```
. ebalfit, baltable noheader notable
Balancing table
------------------------------------------------------------------------------
             |        raw   adjusted  reference     absdif     reldif
```

3. `ebalfit` returns error if perfect balance cannot be achieved, unless option `relax` is specified. The critical value for "perfect balance" can be set using option `btolerance()`. By default, the critical value is set to $10^{-6}$, that is, a solution is considered as balanced if balancing loss, the maximum relative difference between the reweighted means and the target values, is smaller than 0.000001.

| | | | | | |
|---|---|---|---|---|---|
| age | 37.39198 | 26.02386 | 26.02386 | 3.55e-15 | 1.31e-16 |
| education | 10.50566 | 10.52379 | 10.52379 | 0 | 0 |
| black | .4359494 | .8572367 | .8572367 | 1.11e-16 | 5.98e-17 |
| hispanic | .0722888 | .0567015 | .0567015 | 4.16e-17 | 3.94e-17 |
| married | .7481573 | .1944541 | .1944541 | 0 | 0 |
| nodegree | .5295137 | .6608205 | .6608205 | 0 | 0 |
| re74 | 10401.03 | 2230.392 | 2230.392 | 4.55e-13 | 2.04e-16 |
| re75 | 7230.567 | 1682.258 | 1682.258 | 2.27e-13 | 1.35e-16 |

Options `noheader` and `notable` have been specified so that the default header and coefficients table are not displayed again. As is evident, the reweighted means (column "adjusted") perfectly match the target values (column "reference"). Both the absolute difference and the relative difference are essentially zero for all variables.

If we still do not trust this result, we can use `predict` to generate the balancing weights and then construct a balancing table manually:

```
. predict wbal
. table () (treat) [pw=wbal], stat(mean age-re75) nototal
```

| | treat | |
|---|---|---|
| | 0 | 1 |
| age | 26.02386 | 26.02386 |
| education | 10.52379 | 10.52379 |
| black | .8572367 | .8572367 |
| hispanic | .0567015 | .0567015 |
| married | .1944541 | .1944541 |
| nodegree | .6608205 | .6608205 |
| re74 | 2230.392 | 2230.392 |
| re75 | 1682.258 | 1682.258 |

A comparison of the weights from IPW and the weights from entropy balancing reveals that the latter contain more variation:[4]

```
. dstat (cv0) ipw wbal if treat==0
cv0                               Number of obs   =        253
```

| | Coefficient | Std. err. | [95% conf. | interval] |
|---|---|---|---|---|
| ipw | 3.069078 | .2628062 | 2.551501 | 3.586654 |
| wbal | 3.36179 | .4408397 | 2.493591 | 4.22999 |

```
. program DEFF
  1.     syntax varname [if]
  2.     tempvar x2
  3.     quietly generate `x2´ = `varlist´^2
  4.     summarize `x2´ `if´, meanonly
```

4. I use command `dstat` (Jann 2020a), available from the SSC Archive, because it allows computing the CV in the same way as `ebalfit` does. The CV could also be computed using [R] **tabstat**, which applies a slightly different definition (division by $N-1$ rather than $N$ in the variance).

```
  5.     local NX2 = r(sum) * r(N)
  6.     summarize `varlist´ `if´, meanonly
  7.     display as res `NX2´/r(sum)^2
  8. end
. DEFF ipw if treat==0
10.419239
. DEFF wbal if treat==0
12.301634
```

Apparently, the better balance came at the cost of more variation in the weights. Large variation in weights generally reduces statistical efficiency so that weights with lower variation may be preferable. As illustrated below, however, this is not necessarily true for treatment effect analyses because the degree to which the weights balance the data also plays a role for the efficiency of the estimate. Yet, for some applications, for example when using entropy balancing to construct sampling weights, we might want to apply some trimming to the resulting weights to reduce the design effect without sacrificing too much precision in balance.[5]

## 4.2 Computing a treatment effect with corrected standard errors

We now continue estimating the treatment effect on post-treatment earnings. The naive estimate of the ATET (average treatment effect on the treated) is negative:

```
. mean re78 [pw=w0], over(treat)
Mean estimation                           Number of obs = 438
```

| | Mean | Std. err. | [95% conf. | interval] |
|---|---|---|---|---|
| c.re78@treat | | | | |
| 0 | 9104.129 | 758.2113 | 7613.935 | 10594.32 |
| 1 | 6004.657 | 567.8919 | 4888.518 | 7120.796 |

```
. lincom _b[c.re78@1.treat] - _b[c.re78@0bn.treat]
 ( 1)  - c.re78@0bn.treat + c.re78@1.treat = 0
```

| Mean | Coefficient | Std. err. | t | P>\|t\| | [95% conf. | interval] |
|---|---|---|---|---|---|---|
| (1) | -3099.472 | 947.3043 | -3.27 | 0.001 | -4961.311 | -1237.633 |

However, as seen above, the two groups are very different in terms of pre-treatment characteristics. Using IPW or entropy balancing to remove these discrepancies, the treatment effect estimate becomes positive:

```
. mean re78 [pw=ipw], over(treat)
Mean estimation                           Number of obs = 438
```

5. Also see Kranker et al. (2020) who propose penalized CBPS to address this issue (on CBPS see footnote 1).

```
                |       Mean   Std. err.      [95% conf. interval]
----------------+---------------------------------------------------
c.re78@treat    |
              0 |   5088.788   943.9743      3233.493    6944.082
              1 |   6004.657   567.8919      4888.518    7120.796
------------------------------------------------------------------

. lincom _b[c.re78@1.treat] - _b[c.re78@0bn.treat]
 ( 1)  - c.re78@0bn.treat + c.re78@1.treat = 0

------------------------------------------------------------------------------
        Mean | Coefficient  Std. err.      t    P>|t|     [95% conf. interval]
-------------+----------------------------------------------------------------
         (1) |   915.8695    1101.63     0.83   0.406    -1249.282    3081.021
------------------------------------------------------------------------------

. drop ipw
. mean re78 [pw=wbal], over(treat)
Mean estimation                           Number of obs = 438

                |       Mean   Std. err.      [95% conf. interval]
----------------+---------------------------------------------------
c.re78@treat    |
              0 |   4174.016   999.5839      2209.426    6138.605
              1 |   6004.657   567.8919      4888.518    7120.796
------------------------------------------------------------------

. lincom _b[c.re78@1.treat] - _b[c.re78@0bn.treat]
 ( 1)  - c.re78@0bn.treat + c.re78@1.treat = 0

------------------------------------------------------------------------------
        Mean | Coefficient  Std. err.      t    P>|t|     [95% conf. interval]
-------------+----------------------------------------------------------------
         (1) |   1830.641   1149.639     1.59   0.112    -428.8669     4090.15
------------------------------------------------------------------------------

. drop wbal
```

The two effect estimates are not statistically significant, but note that we did not yet correct the standard errors for the fact that the balancing weights are estimated. To do so for the estimate based on entropy balancing, we can use the formulas provided in section 2.5. As inputs we need the influence functions of the entropy balancing coefficients as well as the influence functions of the mean estimates assuming the balancing weights as fixed. The former we can obtain by applying command `predict` after `ebalfit`; the latter we can compute as $\text{IF}^{\hat{\mu}} = \frac{W}{W_S} S_i(x_i - \hat{\mu})$ where $x_i$ is the variable of interest, $S_i$ is an indicator for the analyzed subsample, $W_S$ is the sum of weights in the subsample, and $W$ is the overall sum of weights. In the computations below I omit the leading $W$ because this is how `ebalfit` defines influence functions and because it implies that factor $c$ in the correction formulas will be equal to 1 and can be omitted. To obtain standard errors from influence functions that are scaled in this way command [R] **total** can be used (rather than command [R] **mean**).

```
. ebalfit age-re75 [pw=w0], by(treat)
  (output omitted)
. predict wbal
. predict IFeb*, ifs noalpha // the IF for the constant is not needed
```

```
. summarize re78 if treat==0 [aw=wbal], meanonly
. generate IFy0 = (treat==0) * (re78 - r(mean)) / r(sum_w)
. summarize re78 if treat==1 [aw=wbal], meanonly
. generate IFy1 = (treat==1) * (re78 - r(mean)) / r(sum_w)
. total IFy0 IFy1 [pw=wbal]
Total estimation                              Number of obs = 438
```

| | Total | Std. err. | [95% conf. | interval] |
|---|---|---|---|---|
| IFy0 | -5.68e-13 | 999.5839 | -1964.59 | 1964.59 |
| IFy1 | 5.26e-13 | 567.8919 | -1116.139 | 1116.139 |

Note how `total` applied to the influence functions of the two mean estimates reproduces the standard errors reported by `mean` above. We can now correct the influence functions using the formulas from section 2.5. We only need to correct `IFy0`, the influence function of the mean estimate in the control group because in the treatment group we did not apply any reweighting.

```
. mata:
------------------------------------------------- mata (type end to exit) -----
:     // data
:     grp  = st_data(., "treat")
:     X    = st_data(., "age-re75")
:     IFy0 = st_data(., "IFy0")
:     IFeb = st_data(., "IFeb*")
:     wbal = st_data(., "wbal")
:     w0   = st_data(., "w0")
:     // compute (negative of) G
:     G = colsum(select(wbal :* IFy0 :* X, grp:==0))´
:     // adjust IF
:     st_store(., st_addvar("double", "IFy0c"), wbal :/ w0 :* IFy0 + IFeb * G)
: end
------------------------------------------------------------------------------
```

To compute the corrected standard error of the reweighted mean difference take the total of the difference between the (corrected) influence functions of the two means:

```
. generate IFte = IFy1 - IFy0c
. total IFy0c IFy1 IFte [pw=w0]
Total estimation                              Number of obs = 438
```

| | Total | Std. err. | [95% conf. | interval] |
|---|---|---|---|---|
| IFy0c | 1.42e-13 | 750.6493 | -1475.332 | 1475.332 |
| IFy1 | 5.26e-13 | 567.8919 | -1116.139 | 1116.139 |
| IFte | -8.53e-14 | 906.1741 | -1781.001 | 1781.001 |

```
. drop IF*
```

We see how taking account of the estimated nature of the balancing weights reduces

the standard error of the mean estimate in the control group and also brings down the standard error of the treatment effect estimate, such that the treatment effect estimate is now statistically significant ($t = 1830.6/906.2 = 2.02$, $p = 0.043$).[6]

As mentioned above, entropy balancing is doubly-robust so that applying a regression adjustment model to the reweighted data does not change the estimate of the treatment effect (as long as the same covariates are used in the regression adjustment). I illustrate this here by running [TE] **teffects ra** including the entropy balancing weights:

```
. teffects ra (re78 age-re75) (treat) [pw=wbal], atet

Iteration 0:   EE criterion =  3.671e-22
Iteration 1:   EE criterion =  1.249e-24

Treatment-effects estimation                    Number of obs     =        438
Estimator      : regression adjustment
Outcome model  : linear
Treatment model: none
```

| re78 | Coefficient | Robust std. err. | z | P>\|z\| | [95% conf. | interval] |
|---|---|---|---|---|---|---|
| ATET | | | | | | |
| treat (1 vs 0) | 1830.641 | 905.139 | 2.02 | 0.043 | 56.60155 | 3604.681 |
| POmean | | | | | | |
| treat 0 | 4174.016 | 749.7919 | 5.57 | 0.000 | 2704.45 | 5643.581 |

```
. display _se[ATET:r1vs0.treat] * sqrt(e(N) / (e(N)-1))
906.17408
```

The estimate is still the same and also the standard error is identical even though regression adjustment assumed the balancing weights as fixed (the small difference is because `teffects` divides by $N$ while `total` divides by $N - 1$ when computing the variance). That is, if regression adjustment is used to analyze the balanced data no further correction is needed for the standard errors (as long as the same covariates are used in both models). This may seem remarkable, but it is a direct consequence of the doubly-robust property.

## 4.3 Balancing variances and covariance

In the example above we just balanced the means of the different covariates. To also balance higher moments or covariances we can add powers interactions to the model. Here is an example using a reduced set of covariates:

```
. ebalfit c.age##c.age c.education##c.education 1.black ///
>     c.age#c.education 1.black#c.(age education), by(treat) nolog vsquish

Entropy balancing                               Number of obs     =        438
                                                Wald chi2(8)      =      80.49
                                                Prob > chi2       =     0.0000
```

6. The appendix illustrates how a similar correction can be implemented for IPW.

```
                                                Balancing loss    =  2.532e-15
                                                Loss type         =     reldif
Sample    = 0.treat (253 obs)                   CV of weights     =  1.7253741
Reference = 1.treat (185 obs)                   DEFF of weights   =  3.9769159
```

| | Coefficient | Robust std. err. | z | P>\|z\| | [95% conf. | interval] |
|---|---|---|---|---|---|---|
| age | -.3454953 | .1819311 | -1.90 | 0.058 | -.7020737 | .0110831 |
| c.age#c.age | -.0007966 | .0017921 | -0.44 | 0.657 | -.004309 | .0027159 |
| education | -.6033715 | .8552482 | -0.71 | 0.481 | -2.279627 | 1.072884 |
| c.education# c.education | -.0240512 | .0360925 | -0.67 | 0.505 | -.0947912 | .0466887 |
| 1.black | -3.866887 | 2.519995 | -1.53 | 0.125 | -8.805986 | 1.072212 |
| c.age# c.education | .0232092 | .0105351 | 2.20 | 0.028 | .0025608 | .0438577 |
| black#c.age 1 | .0905916 | .0420067 | 2.16 | 0.031 | .00826 | .1729231 |
| black# c.education 1 | .2766195 | .1750993 | 1.58 | 0.114 | -.0665689 | .6198078 |
| _cons | 11.74866 | 6.392829 | 1.84 | 0.066 | -.7810508 | 24.27838 |

```
. predict wbal2
. summarize age education black if treat==0 [iw=wbal2]
```

| Variable | Obs | Weight | Mean | Std. dev. | Min | Max |
|---|---|---|---|---|---|---|
| age | 253 | 185 | 25.81622 | 7.155019 | 18 | 55 |
| education | 253 | 185 | 10.34595 | 2.01065 | 0 | 17 |
| black | 253 | 185 | .8432432 | .3645579 | 0 | 1 |

```
. summarize age education black if treat==1 [iw=wbal2]
```

| Variable | Obs | Weight | Mean | Std. dev. | Min | Max |
|---|---|---|---|---|---|---|
| age | 185 | 185 | 25.81622 | 7.155019 | 17 | 48 |
| education | 185 | 185 | 10.34595 | 2.01065 | 4 | 16 |
| black | 185 | 185 | .8432432 | .3645579 | 0 | 1 |

```
. corr age education black if treat==0 [aw=wbal2]
(sum of wgt is 185)
(obs=253)
```

| | age | educat~n | black |
|---|---|---|---|
| age | 1.0000 | | |
| education | -0.0080 | 1.0000 | |
| black | 0.0535 | -0.0368 | 1.0000 |

```
. corr age education black if treat==1 [aw=wbal2]
(sum of wgt is 185)
(obs=185)
```

| | age | educat~n | black |
|---|---|---|---|
| age | 1.0000 | | |
| education | -0.0080 | 1.0000 | |
| black | 0.0535 | -0.0368 | 1.0000 |

We see that means, standard deviations (and variances), as well as correlations (and covariances) have been perfectly balanced. Alternatively, option `targets()` can be used

to generate the necessary terms automatically. `ebalfit` will then expand the variable list accordingly, taking account of the types of the variables (e.g., no terms for variances of categorical variables will be included as balancing the mean of a 0/1 variable also balances its variance):

```
. ebalfit age education 1.black, by(treat) targets(variance covariance) ///
>     nolog vsquish
Entropy balancing                               Number of obs     =        438
                                                Wald chi2(8)      =      80.49
                                                Prob > chi2       =     0.0000
                                                Balancing loss    =  2.532e-15
                                                Loss type         =     reldif
Sample    = 0.treat (253 obs)                   CV of weights     =  1.7253741
Reference = 1.treat (185 obs)                   DEFF of weights   =  3.9769159
```

| | Coefficient | Robust std. err. | z | P>\|z\| | [95% conf. | interval] |
|---|---|---|---|---|---|---|
| age | -.3454953 | .1819311 | -1.90 | 0.058 | -.7020737 | .0110831 |
| c.age#c.age | -.0007966 | .0017921 | -0.44 | 0.657 | -.004309 | .0027159 |
| education | -.6033715 | .8552482 | -0.71 | 0.481 | -2.279627 | 1.072884 |
| c.education# c.education | -.0240512 | .0360925 | -0.67 | 0.505 | -.0947912 | .0466887 |
| 1.black | -3.866887 | 2.519995 | -1.53 | 0.125 | -8.805986 | 1.072212 |
| c.age# c.education | .0232092 | .0105351 | 2.20 | 0.028 | .0025608 | .0438577 |
| black#c.age 1 | .0905916 | .0420067 | 2.16 | 0.031 | .00826 | .1729231 |
| black# c.education 1 | .2766195 | .1750993 | 1.58 | 0.114 | -.0665689 | .6198078 |
| _cons | 11.74866 | 6.392829 | 1.84 | 0.066 | -.7810508 | 24.27838 |

Manually typing out the higher-oder and interaction terms is only needed if one wants to balance a subset of the higher moments and covariances (e.g. the covariance between `age` and `education`, but not between `black` and the other variables).

## 4.4 Adjusting a sample to population values

Assume that you know from census data that the mean age is 30, the mean of education is 10, and the distribution of ethnicities is 50% white, 40% black, 10% hispanic. The `population()` option can be used (instead of option `by()`) to balance the sample to these values:

```
. ebalfit age education black hispanic, population(30 10 .4 .1) baltable nolog
Entropy balancing                               Number of obs     =        438
                                                Wald chi2(4)      =      92.37
                                                Prob > chi2       =     0.0000
                                                Balancing loss    =  1.547e-14
                                                Loss type         =     reldif
                                                CV of weights     =  .68869277
Population size = 438                           DEFF of weights   =  1.4742977
```

| | Coefficient | Robust std. err. | z | P>\|z\| | [95% conf. | interval] |
|---|---|---|---|---|---|---|
| age | -.0392513 | .0058811 | -6.67 | 0.000 | -.0507781 | -.0277246 |
| education | -.1730964 | .0264454 | -6.55 | 0.000 | -.2249285 | -.1212643 |
| black | -1.206872 | .1398756 | -8.63 | 0.000 | -1.481023 | -.9327208 |
| hispanic | -.3027049 | .2401386 | -1.26 | 0.207 | -.7733679 | .167958 |
| _cons | 3.614511 | .4453534 | 8.12 | 0.000 | 2.741634 | 4.487388 |

Balancing table

| | raw | adjusted | reference | absdif | reldif |
|---|---|---|---|---|---|
| age | 31.75342 | 30 | 30 | 4.83e-13 | 1.56e-14 |
| education | 10.58904 | 10 | 10 | 1.40e-13 | 1.28e-14 |
| black | .5821918 | .4 | .4 | 3.50e-15 | 2.50e-15 |
| hispanic | .0639269 | .1 | .1 | 1.05e-15 | 9.59e-16 |

The balancing table illustrates that the reweighted sample data perfectly reproduces the population values. We did not specify a population size, so `ebalfit` normalized the sum of balancing weights to the sample size. Assume that the size of the population is 1.36 million. We could normalize the weights to this target sum as follows:

```
. ebalfit age education black hispanic, population(1.36e6: 30 10 .4 .1) nolog
Entropy balancing                               Number of obs     =        438
                                                Wald chi2(4)      =      92.37
                                                Prob > chi2       =     0.0000
                                                Balancing loss    =  1.547e-14
                                                Loss type         =     reldif
                                                CV of weights     =  .68869277
Population size = 1,360,000                     DEFF of weights   =  1.4742977
```

| | Coefficient | Robust std. err. | z | P>\|z\| | [95% conf. | interval] |
|---|---|---|---|---|---|---|
| age | -.0392513 | .0058811 | -6.67 | 0.000 | -.0507781 | -.0277246 |
| education | -.1730964 | .0264454 | -6.55 | 0.000 | -.2249285 | -.1212643 |
| black | -1.206872 | .1398756 | -8.63 | 0.000 | -1.481023 | -.9327208 |
| hispanic | -.3027049 | .2401386 | -1.26 | 0.207 | -.7733679 | .167958 |
| _cons | 11.65529 | .4453534 | 26.17 | 0.000 | 10.78241 | 12.52816 |

The target values do not necessarily need to be true values from a population. We can also use entropy balancing to construct as-if scenarios by setting the targets to theoretically interesting values, as long as the targets are not too far away from the center of the data such that no balancing solution exists.

# 5 Conclusions

Entropy balancing is a powerful alternative to other reweighting techniques such as inverse probability weighting based on logistic regression. In this paper I defined the

model, derived influence functions for the parameters of the model, and illustrated how consistent standard errors can be obtained for statistics based on the reweighted data. I further presented software that implements the discussed methods.

The software provides a convenient tool for estimating balancing weights and generating influence functions, but adjusting statistical inference for reweighted statistics still requires a good understanding of the problem at hand and some programming skills on the side of the user. Based on the results presented in this paper, entropy balancing could be integrated into other estimation commands as a preprocessing device, such that reweighted estimates with consistent statistical inference would be readily available for a variety of applications without the need to write lengthy code. Some steps in this direction have already been taken. Command `dstat`, available from the SSC Archive, offers a `balance()` option that applies reweighting to a large collection of summary statistics (Jann 2020a). In fact, the treatment effect estimate in section 4.2 can be replicated by `dstat` in a single line of code:

```
. dstat re78 [pw=w0], over(treat, contrast) balance(eb:age-re75, reference(1))
Difference in mean                              Number of obs     =        438
                                                Contrast          =    0.treat
                                                Balancing:
                                                          method  =         eb
                                                       reference  =    1.treat
                                                        controls  = e(balance)
```

| re78 | Coefficient | Std. err. | t | P>\|t\| | [95% conf. | interval] |
|---|---|---|---|---|---|---|
| 1.treat | 1830.641 | 906.1741 | 2.02 | 0.044 | 49.64029 | 3611.643 |

Similar support for reweightig is provided in `reldist`, a command for relative distribution analysis (Jann 2020c). Furthermore, the matching and reweighting command `kmatch` (Jann 2017) supports entropy balancing, albeit without specific adjustment of statistical inference. Since regression adjustment after entropy balancing yields consistent standard errors, however, command `kmatch` can still be used to obtain valid results. Here is a replication of the treatment effect from section 4.2 using `kmatch`. The trick is to include the full vector of covariates also in the outcome equation.

```
. kmatch eb treat age-re75 (re78=age-re75) [pw=w0], nomtable att
(fitting balancing weights ... done)
Entropy balancing                                       Number of obs = 438
                                                Balance tolerance =     .00001
Treatment   : treat = 1
Targets     : 1
Covariates  : age education black hispanic married nodegree re74 re75
RA equations: re78 = age education black hispanic married nodegree re74 re75 ...
Treatment-effects estimation
```

| re78 | Coefficient | Std. err. | t | P>\|t\| | [95% conf. | interval] |
|---|---|---|---|---|---|---|
| ATT | 1830.641 | 906.1741 | 2.02 | 0.044 | 49.64029 | 3611.643 |

Implementing entropy balancing as a reweighting and standardization option may also be useful for various other types of estimators, but such work still has to done. Another topic that deserves further investigation whether the efficiency of reweighted statistics based on entropy balancing can further be improved by optimized trimming techniques, and how this would affect statistical inference. See Kranker et al. (2020) for work that goes in this direction

# 6 Appendix: Correction of IPW standard errors

For IPW we can adjust the standard errors of statistics based on the reweighted data in a similar way as for entropy balancing. In fact, working through the details (see Jann 2020c) shows that for IPW based on a logit model the correction has the exact same form as for entropy balancing. We simply need to replace the entropy balancing influence functions by the influence functions of the parameters of the logit model. A convenient way to compute influence functions for maximum likelihood models makes use of the scores and the model-based variance matrix (see Jann 2020b). The correction goes as follows:

```
. logit treat age-re75 [pw=w0]
  (output omitted)
. predict pscore if treat==0, pr
(185 missing values generated)
. generate ipw = w0 * cond(treat==0, pscore/(1-pscore), 1)
. predict score, score
. forvalues i=1/9 {
  2.     quietly generate IFipw`i´ = .
  3. }
. mata:
------------------------------------------------- mata (type end to exit) -------
:     X     = st_data(.,"age-re75")
:     X     = X, J(rows(X), 1, 1) // add constant
:     score = st_data(.,"score")
:     Ginv  = st_matrix("e(V_modelbased)")
:     st_view(IF=., ., "IFipw*")
:     IF[.,.] = (X :* score) * Ginv´
: end
--------------------------------------------------------------------------------

. drop score
. drop IFipw9 // drop the IF for the constant; it is not needed
. summarize re78 if treat==0 [aw=ipw], meanonly
. generate IFy0 = (treat==0) * (re78 - r(mean)) / r(sum_w)
. summarize re78 if treat==1 [aw=ipw], meanonly
. generate IFy1 = (treat==1) * (re78 - r(mean)) / r(sum_w)
. mata:
------------------------------------------------- mata (type end to exit) -------
:     // data
:     grp   = st_data(., "treat")
```

```
:     X     = st_data(., "age-re75")
:     IFy0  = st_data(., "IFy0")
:     IFipw = st_data(., "IFipw*")
:     ipw   = st_data(., "ipw")
:     w0    = st_data(., "w0")
:     // compute (negative of) G
:     G = colsum(select(ipw :* IFy0 :* X, grp:==0))´
:     // adjust IF
:     st_store(., st_addvar("double", "IFy0c"), ipw :/ w0 :* IFy0 + IFipw * G)
: end
```

```
. generate IFte = IFy1 - IFy0c
. total IFy0c IFy1 IFte [pw=w0]
Total estimation                              Number of obs = 438
```

| | Total | Std. err. | [95% conf. | interval] |
|---|---|---|---|---|
| IFy0c | -6.88e-08 | 943.3969 | -1854.159 | 1854.159 |
| IFy1 | 5.26e-13 | 567.8919 | -1116.139 | 1116.139 |
| IFte | 6.88e-08 | 1071.73 | -2106.385 | 2106.385 |

```
. drop IF*
```

The effect of the correction is less pronounced than for entropy balancing. This is related to the finding that conditioning on the estimated propensity score is more efficient than conditioning on the true propensity score, because random imbalance is partially removed. For entropy balancing this efficiency gain is stronger than for IPW because entropy balancing completely removes random imbalance.

# 7 References

Dehejia, R. H., and S. Wahba. 1999. Causal Effects in Non-Experimental Studies: Reevaluating the Evaluation of Training Programs. *Journal of the American Statistical Association* 94(448): 1053–1062.

Hainmueller, J. 2012. Entropy Balancing for Causal Effects: A Multivariate Reweighting Method to Produce Balanced Samples in Observational Studies. *Political Analysis* 20(1): 25–46.

Hainmueller, J., and Y. Xu. 2011. ebalance: Stata module to perform Entropy reweighting to create balanced samples. Statistical Software Components S457326. Available from https://ideas.repec.org/c/boc/bocode/s457326.html.

———. 2013. ebalance: A Stata Package for Entropy Balancing. *Journal of Statistical Software* 54(7): 1–18.

Imai, K., and M. Ratkovic. 2014. Covariate balancing propensity score. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 76(1): 243–263.

Imbens, G. W., and J. M. Wooldridge. 2009. Recent Developments in the Econometrics of Program Evaluation. *Journal of Economic Literature* 47(1): 5–86.

Jann, B. 2005. moremata: Stata module (Mata) to provide various functions. Statistical Software Components S455001. Available from http://ideas.repec.org/c/boc/bocode/s455001.html.

———. 2017. kmatch: Stata module for multivariate-distance and propensity-score matching, including entropy balancing, inverse probability weighting, (coarsened) exact matching, and regression adjustment. Statistical Software Components S458346. Available from https://ideas.repec.org/c/boc/bocode/s458346.html.

———. 2020a. DSTAT: Stata module to compute summary statistics and distribution functions including standard errors and optional covariate balancing. Statistical Software Components S458874,. Available from https://ideas.repec.org/c/boc/bocode/s458874,.html.

———. 2020b. Influence functions continued. A framework for estimating standard errors in reweighting, matching, and regression adjustment. University of Bern Social Sciences Working Papers 35. Available from https://ideas.repec.org/p/bss/wpaper/35.html.

———. 2020c. Relative distribution analysis in Stata. University of Bern Social Sciences Working Papers 37. Available from https://ideas.repec.org/p/bss/wpaper/37.html.

Kish, L. 1965. *Survey Sampling*. New York: Wiley.

Kranker, K. 2019. psweight: IPW- and CBPS-type propensity score reweighting, with various extensions. Statistical Software Components S458657. Available from https://ideas.repec.org/c/boc/bocode/s458657.html.

Kranker, K., L. Blue, and L. Vollmer Forrow. 2020. Improving Effect Estimates by Limiting the Variability in Inverse Propensity Score Weights. *The American Statistician* .

LaLonde, R. J. 1986. Evaluating the Econometric Evaluations of Training Programs with Experimental Data. *The American Economic Review* 76(4): 604–620.

Valliant, R., J. A. Dever, and F. Kreuter. 2013. *Practical Tools for Designing and Weighting Survey Samples*. New York: Springer.

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