ANES Weighting Algorithm
A Description

Josh Pasek
Stanford University[*]

March 15, 2010

The Need For A New Algorithm

Survey weighting is methodological a black box. While survey research firms are often acutely aware of the large number of arbitrary decisions that go into the production of survey weights, they often provide researchers with a single vector of weights that represent only one such set of decisions. Researchers have typically been removed from the process. While most practitioners have been taught to include weighted data in their analyses, few know where those weights come from or what kinds of decisions are involved. This means that the weights constructed are rarely well attuned to the researchers' particular questions of interest.

The challenge posed by survey weighting has recently been examined by a panel of experts convened by the American National Election Study. In a report based on the conclusions of that panel, DeBell and Krosnick (2009) propose a means of standardizing the survey weighting procedure so as to avoid potential methodological pitfalls. In this regard, they identify a number of decisions that individuals creating a weighting metric must make and propose a default procedure that will result in fairly accurate estimates.

While the ANES Report takes a substantive step toward broadening the capacity of individual researchers wishing to create survey weights, existing software is not well attuned to the many

[*]Please direct all correspondence to: Josh Pasek, Stanford University, Department of Communication, 450 Serra Mall, Stanford, CA 94305. Email: josh@joshpasek.com.

suggestions in the memo and is often opaque with regard to specific methodological decisions. For instance, the widely used $Survey$ package in R does not allow researchers to cap weights at each iteration, one of the suggestions in the memo, and does not help researchers determine which variables should be selected for weighting. Similar challenges can be found in other popular software alternatives.

To this end, there is a large potential benefit from a software package designed to openly and directly address the potential decisions that a researcher may wish to make. This paper outlines the creation of the ANES Weighting Algorithm (AWA), a weighting package in R based on the ANES Report which is designed to allow researchers to simply and easily customize the series of weights they wish to generate.

Basics Of The ANES Weighting Algorithm

As advised in the ANES Report, the AWA uses an iterative, multiplicative, raking model to generate weights. In this kind of design, the survey marginals for a given set of variables are compared to known population marginals for each variable of interest. In each iteration of a single weighting variable, a vector of weights is adjusted such that the sum of the weights for a particular variable account for the same proportion of the overall sample as the known population marginal.

To better illustrate this process, one can imagine a sample of the American public in 2008 which happened to be 45% male and 55% female. The actual 2008 American population (according to the weighted March 2008 Current Population Survey) was instead 48% male and 52% female. Each man in the sample would then have his weight vector multiplied by $\frac{48}{42}$, while each woman in the sample would have her weight multiplied by $\frac{52}{55}$. The sum of the weighted vectors for men and women would then be equivalent to the numbers for the actual American population.

If the second variable used for weighting was race, the AWA would use the weights vector created from gender to determine the weighted percentage of individuals of each race. To extend our example, we might imagine that 80% of our sample was White Non-Hispanic, 7% of our sample was Black Non-Hispanic, 7% of our sample was Hispanic, and 6% of our sample was some other

race. The actual distributions were 69% White, 11% Black, 13% Hispanic, and 6% Other. The new weights for each individual would include the old weights multiplied by the ratio between the sample marginals and the population marginals. Hence, the new weights for a White individual would be equivalent to $old\_weight \times \frac{69}{80}$.

This basic procedure would continue until the sample had been adjusted for all of the variables – a single iteration. The algorithm then repeats this process of adjusting for all of the variables until one of three things happens: (1) The algorithm fully converges such that the weighted sample marginal percentages for all categories perfectly match the population marginals, (2) the algorithm is not able to converge any further even though the weighted sample marginals do not perfectly match, or (3) the algorithm is completely unable to converge because the specifications provided could not be used.[1]

Implications of the Log-Linear Model Underlying Weighting

While the process for producing raking weights is relatively straightforward, a variety of assumptions are implicit in the weighting model. As with most attempts to correct for some level of survey non-response, raking depends fundamentally on the assumption that non-response for any given outcome (y) can be treated as Missing At Random (MAR) conditional on a set of varaibles ($\mathbf{X}$). If ω defines the sample of individuals selected, k indexes individuals, n indexes the size of the sample, and N is the population size, the basic assumption could be illustrated as follows:

$$\frac{1}{n}\sum_{k\in\omega}(y_k|\mathbf{X_k}) = \frac{1}{N-n}\sum_{k\notin\omega}(y_k|\mathbf{X_k}) \tag{1}$$

$$Pr(\bar{y}_{k\in\omega}|\mathbf{X_{k\in\omega}}) = Pr(\bar{y}_{k\notin\omega}|\mathbf{X_{k\notin\omega}}) \tag{2}$$

[1]The AWA provides a warning if full convergence is not possible (option 2) and stops with an error if the specifications are impossible to match (option 3).

This assumption, whether tenable or not, underlies every single attempt to correct for survey non-response. But the specific nature of any given correction for non-response introduces additional implicit assumptions. In the specific case of raking, the iterative proportional fitting algorithm used here minimizes the KL Divergence between sample marginals and the specified targets for each subcategory of X. In conforming to the target marginals, iterative proportional fitting coverges geometrically. A fuller discussion of the underlying model falls outside the purpose of this memorandum. Ireland and Kullback first outlined this process in their seminal 1968 paper "Contingency tables with given marginals" in *Biometrika* and describe the mathematical processes involved.

The use of raking introduces two additional and important assumptions about the nature of missingness in the data. First, because the $\mathbf{X}$ variables are categorical, rather than continuous, researchers must be comfortable with the presumption that variation in non-response is MAR conditional on the categories used, not just the overall variable. This may be more comfortable than a pure linear assumption for certain uses, but nonetheless depends on individuals within categories behaving in a sufficiently similar way. A researcher dividing respondents into age categories for raking might choose to classify individuals between the ages of 25 and 29 either with individuals aged 18-24 or with those 30-34. Classifying this group with 18-24 year-olds rests of the assumption that the group of individuals aged 18-30 are similar with regard to the outcomes of interest. In contrast, classifying this group with 30-34 year-olds presumes similarity across the 25-34 age range. While technically somewhat arbitrary, the distinction is substantive.

The second major assumption implicit in the raking model is that variation in the outcome due to any given $\mathbf{X}$ variable is orthogonal to variation in the outcomes due to any other $\mathbf{X}$ variable. Raking, as a procedure, will not capture correlations between $\mathbf{X}$ variables unless the cross-classifications of those variables are themselves included in the set of $\mathbf{X}$s. For instance, the well known relationship between gender and age – namely that men outnumber women among young people, but that women outnumber men in old age – will not be reproduced by raking to the marginals of the two categories. This is not a substantive issue if the composition of the sample matches society on the features for

which raking is used. Hence, if the gender-age relationship is appropriately captured in the initial sample, the raking procedure will not likely undermine the correlation.

AWA Capabilities

Capping

In addition to the basic raking function of the AWA, the algorithm also provides the ability to iteratively cap weights. At the end of each iteration of the algorithm, any weights larger than a user-set cap will be automatically set to equal that cap. Weights are then adjusted to have a mean of 1 (by dividing the weights by their mean after the cap). This capping procedure is repeated until the highest weight is no larger than the cap plus .0001. After each capping procedure, the next iteration of the raking procedure proceeds using the capped weights as an initial weighting vector. If the next iteration of raking raises the largest weight above the cap, the procedure is followed again by an additional round of capping. By default, the cap is set to an arbitrarily high number.

Base Weights

The AWA procedure is also able to accept a vector of base weights from which the weighting procedure will begin. The base weights can be used to adjust for known differences between the sampling frame and the general public or for other adjustments necessary before the algorithm is run.

Identification of Discrepant Variables

Raking on a large number of variables can quickly increase the variance of the weights for the dataset and lead to inaccurate estimates. For this reason, the ANES Report suggests that researchers focus on the demographic variables which are most deviant from known population marginals. To establish the level of deviation, the AWA provides six options for identifying the discrepancy between sample marginals for variables and five options for selecting variables for inclusion in

the raking procedure. Variables can be chosen based on (1) the sum of the deviations between the sample marginals and the target marginals, (2) the largest single deviation between the sample marginals and the target marginals, (3) the average deviation between the sample marginals and the target marginals, (4) the sum of the squared deviations between the sample marginals and target marginals, (5) the largest squared deviation between the sample marginals and the target marginals[2], and (6) the average squared deviation between the sample marginals and the target marginals.

Users of the AWA can then specify one of five metrics for choosing variables that will be included: (1) including all variables, (2) including variables where the sum of the differences is larger than a user-specified amount, (3) including a set number of variables with the largest deviations, (4) including either a set number of variables or however many variables have deviations of larger than a user-specified amount, whichever includes more variables, or (5) including however many variables have deviations of larger than a user-specified amount, unless that number is larger than a set number of variables, in which case that set number is instead chosen. In all cases, the variables identified as most discrepant are chosen first.

Filtering

Raking can be run on only a subsection of the datset using a filtering command.

Missing Data

The AWA algorithm can handle missing data and does so by including missing data as a separate weighting category for each variable iteration. When weighting on a variable, weights are only adjusted for individuals who do not have missing data on that specific variable. These individuals are still included for every other weighting category, however, and are assigned a weight based on all non-missing data. Cases missing all variables will retain a weight of 1.

[2]Note that this is equivalent to (2).

Missing values can be set either to "NA" or to equal a value of one greater than the number of categories specified by the list element for any given variable. In the gender example above, an individual coded with a gender value of 3 would be treated as missing.

Using the AWA

```
anesrake(inputter, dataframe, caseid, weightvec = NULL, cap = 5,
verbose = FALSE, maxit = 1000, type = "pctlim", pctlim = 5,
nlim = 5, filter = 1, choosemethod = "total", iterate = TRUE)
```

inputter The `inputter` command should contain a list of all target values for the raking procedure. Each list element in `inputter` should be a vector corresponding to the weighting targets for a single variable. Hence, the vector enumerating the weighting targets for a variable with 2 levels should be of length 2, while a vector enumerating the weighting targets for a variable with 5 levels should be of length 5. List elements in inputter should be named according to the variable that they will match in the corresponding dataset. Hence, a list element enumerating the proportion of the sample that should be of each gender should be labeled "female" if the variable in `dataframe` is also titled "female."

`inputter` elements must be vectors and can be of class numeric, or factor and must match the class of the corresponding variable in `dataframe`. Logical variables in `dataframe` can be matched to a numeric vector of length 2 and ordered with the `TRUE` target as the first element and the `FALSE` target as the second element. Targets for factors must be labeled to match every level present in the dataframe (e.g. a variable with 2 age groups "under40" and "over40" should have elements named "under40" and "over40" respectively). `anesrake` attempts to conform any unrecognized types of vectors to `class(numeric)`. Weighting targets can be entered either as an N to be reached or as a percent for any given variable. Targets can be either proportions (ideal) or the number of individuals in the population in each target category (N). Totals of greater than 1.5 for any given list element are treated as Ns, while values of less than 1.5 are treated as percentages.

A list of this sort might appear as follows:

```
inputter
$gender
     Male   Female
[1] 0.48   0.52

$race
    White   Black  Hispanic  Other
[1] 0.69    0.11   0.13      0.06
```

dataframe The `inputter` command identifies a `data.frame` object of the data to be weighted. The data.frame must contain all of the variables that will be used in the weighting process and those variables must have the same names as are present in the `inputter` list element.

caseid The `caseid` command identifies a unique case identifier for each individual in the dataset. If filters are to be used, the resulting list of weights will be a different length from the overall `dataframe`. `caseid`is included in the output so that weights can be matched to the dataset of relevance. `caseid` must be of a length matching the number of cases in `dataframe`.

weightvec `weightvec` is an optional input if some kind of base weights, stratification correction, or other sampling probability of note that should be accounted for before weighting is conducted. If defined, `weightvec` must be of a length equivalent to the number of cases in the `dataframe`. If undefined, `weightvec` will be automatically seeded with a vector of 1s.

cap `cap` defines the maximum weight to be used. `cap` can be defined by the user with the command `cap=X`, where `X` is any value above 1 at which the algorithm will cap weights. If `cap` is set below 1, the function will return an error. If `cap` is set between 1 and 1.5, the function will return a warning that the low cap may substantially increase the amount of time required for weighting. In the absence of a user-defined cap, the AWA defaults to a starting value of 5 in line with DeBell and Krosnick, 2009. For no cap, `cap` simply needs to be set to an arbitrarily high number.

verbose Users interested in seeing the progress of the AWA can set `verbose` to equal `TRUE`. The algorithm will then inform the user of the progress of each raking and capping iteration.

maxit Users can set a maximum number of iterations for the function should it fail to converge using `maxit=X`, where `X` is the maximum number of iterations. The default is set to 1000.

type `type` identifies which manner of variable identification should be used to select weighting variables. Five options are available: type=c("nolim", "pctlim", "nlim", "nmin", "nmax"). If `type="nolim"`, all variables specified in `inputter` will be included in the weighting procedure. If `type="pctlim"` (DEFAULT), the variable selection algorithm will assess which variables have distributions that deviate from their targets by more than the amount specified by the `pctlim` command using the method `choosemethod`. If `type="nlim"`, the variable selection algorithm will use the number of varibles specified by `nlim`, choosing the most discrepant variables as identified by the `choosemethod` command. If `type="nmin"`, the variable selection algorithm will use at least `nlim` variables, but will include more if additional variables are off by more than `pctmin` (all identified using `choosemethod`). If `type="nmax"`, the variable selection algorithm will use no more than `nlim` variables, but will only use that many variables if at least that many are off by more than `pctlim` (all identified using `choosemethod`).

pctlim `pctlim` is the discrepancy limit for selection. Variable selection will only select variables that are discrepant by more than the amount specified. `pctlim` can be specified either in percentage points (5 is 5 percent) or as a decimal (.05 is 5 percent). The algorithm assumes that a decimal is being used if `pctlim<1`. Hence researchers interested in a discrepancy limit of half a percent would need to use `pctlim=.005`.

nlim `nlim` is the number of variables to be chosen with variable selection method.

filter `filter` is a vector of 1 for cases to be included in weighting and 0 for cases that should not be included. The `filter` vector must have the same number of cases as the `dataframe`. In

the absence of a user-defined `filter`, the AWA defaults to a starting value of 1 (inclusion) for all individuals.

choosemethod `choosemethod` is the method for choosing most discrepant variables. Six options are available: choosemethod=c("total", "max", "average", "totalsquared", "maxsquared", "averagesquared"). If `choosemethod="total"`, variable choice is determined by the sum of the differences between actual and target values for each prospective weighting variable. If `choosemethod="max"`, variable choice is determined by the largest individual difference between actual and target values for each prospective weighting variable. If `choosemethod="average"`, variable choice is determined by the mean of the differences between actual and target values for each prospective weighting variable. If `choosemethod="totalsquared"`, variable choice is determined by the sum of the squared differences between actual and target values for each prospective weighting variable. If `choosemethod="maxsquared"`, variable choice is determined by the largest squared difference between actual and target values for each prospective weighting variable (note that this is identical to `choosemethod="max"` if the selection type is `nlim`). If `choosemethod="averagesquared"`, variable choice is determined by the mean of the squared differences between actual and target values for each prospective weighting variable.

iterate `iterate` is a logical variable for how raking should proceed if type=c("pctlim", "nmin", "nmax") conditions. If `iterate=TRUE`, `anesrake` will check whether any variables that were not used in raking deviate from their targets by more than `pctlim` percent. When this is the case, raking will be rerun using the raked weights as seeds (`weightvec`) with additional varibles that meet this qualification after raking included as well. For the `type="nmax"` condition, this will only occur if `nlim` has not been met.

convcrit `convcrit` is the criterion for convergence. The raking algorithm is determined to have converged when an the most recent iteration removes is less than a `convcrit` percentage improvement over the prior iteration.

Example of Algorithm Use

```
load("../datasets/anes04.rdata")

anes04$caseid <- 1:length(anes04$age)

anes04$agecats <- cut(anes04$age, c(25,35,45,55,65))
levels(anes04$agecats) <- c("age1824", "age2534", "age3544",
          "age4554", "age5564", "age6599")

marriedtarget <- c(.4, .6)

agetarg <- c(.10, .15, .17, .23, .22, .13)
names(agetarg) <- c("age1824", "age2534", "age3544",
          "age4554", "age5564", "age6599")

targets <- list(marriedtarget, agetarg)

names(targets) <- c("married", "agecats")

outsave <- anesrake(targets, anes04, caseid=anes04$caseid,
          verbose=TRUE)

write.csv(print(outsave), "outsave.csv")

summary(outsave)
```