Paper 275-28

SAS® Macros and Tools for Working with Weighted Logistic Regression Models That Use Survey Data

David Izrael, Annabella A. Battaglia, David C. Hoaglin, and Michael P. Battaglia, Abt Associates Inc., Cambridge, MA

1. Introduction

When the data come from a survey with weights, working with logistic regression models often involves a number of challenges. We present SAS macros to facilitate three key steps: 1) selecting a model, 2) comparing the predictive ability of several models, and 3) assessing variability. For stepwise selection of a model, one macro augments the SAS output by displaying two information criteria, the Akaike Information Criterion and the Schwartz Criterion, at each step and identifying a stopping point for the stepwise model. One measure of predictive ability is the statistic *c*, which PROC LOGISTIC calculates in unweighted form. We present a SAS macro for *c* that takes the survey weights into account. The resampling method known as the bootstrap provides a framework for assessing variability of models and estimates. For comparing the models, we needed to assess the differences in *c* against the variation in the data. We developed a set of bootstrap samples, and those allowed us to calculate a bootstrap standard error for the difference in *c*. With this brief overview we now describe these three steps and then discuss the application of them in our study.

2. Selecting a Model

The building of a logistic regression model often proceeds stepwise, adding one explanatory variable at each step. Customarily, the variable added is the one that produces the greatest increase in the (binomial) likelihood. These steps can be allowed to continue until all available explanatory variables are in the model; but beyond some point the contributions are often minor, reflecting mainly noise in the data. A variety of criteria have been proposed for determining when to stop adding variables to the model or for selecting a suitably parsimonious model. One basic strategy uses the likelihood-ratio test (based on the chi-squared distribution of –2 times the logarithm of the likelihood, under appropriate assumptions) and requires that each variable added be significant at a specified level (e.g., SLENTRY = 0.05, the default value).

Another approach starts with –2 log *L* and adds a "penalty term" that is an increasing function of the number of parameters in the model. Without the penalty term, maximizing the likelihood (*L*) is equivalent to minimizing –2 log *L*. Thus the penalty term aims to guide the selection toward a more-parsimonious model by causing the minimum of the modified log-likelihood to occur at a smaller number of parameters. The two best-known instances of this approach are the Akaike Information Criterion (AIC) and the Bayesian Information Criterion (BIC), also known as the Schwartz Criterion (SC). If s_i is the number of parameters associated with the explanatory variables that are in the model at Step i and k is the total number of response levels minus one ($k = 1$, if the response is binomial), then

$$\mathrm{AIC}_i = -2 \log L_i + 2(k+s_i) .$$

And if n is the number of observations, then

$$\mathrm{SC}_i = -2 \log L_i + (k+s_i)\log(n).$$

The above formulas are used by PROC LOGISTIC to calculate AIC and SC. Because these criteria have advantages and disadvantages, we usually prefer to display both AIC and SC for each step, until the likelihood-ratio test adds no further variables to the model. Shtatland et al. (2000) discuss some properties of AIC and SC and recommend their complementary use.

Using SAS to select a weighted logistic regression model

We use stepwise model selection to calculate, tabulate, and plot AIC and SC at each step in a user-friendly format, and to determine the step at which SC reaches its minimum. In contrast to the conventional significance level of entry (SLENTRY or SLE) equal to 0.05, we use (following Shtatland et al. 2000) SLE high enough (0.95, for instance) to allow the stepwise selection to continue until the model includes all explanatory variables of interest.

The macro *%SELECT* presented below uses ODS facilities to retrieve the data related to AIC and SC along with other statistics at each step of the stepwise procedure, calculates and marks the recommended stopping point as the step where SC reaches its minimum, and also calculates the change in AIC and SC at each step. The macro consolidates all analytical data into one customized table and generates AIC and SC plots as functions of STEP (Table A1 and Figure A1 in the Appendix).

The macro is presented in Exhibit 1. It includes extensive comments and hardly needs further explanation.

Exhibit 1. Macro *%SELECT*

```
%macro select ( ds =        /* input data set */
                outds =     /* output data set */
```

```
            weight=      /* survey weight */
            model =      /* all predictors must
                            be categorical */
            depvar=      /* dependent variable */
            sle   =      /* probability for entry */
           );

ods listing close;
ods output FitStatistics=_crit
ModelBuildingSummary=_summary;  /* define data sets for
                                   ODS output */

proc logistic descending data=&ds;    /* proc logistic */
weight &weight/norm;                  /* weight normalized */
class &model;                         /* all variables
                                         categorical */
 model &depvar = &model
/selection= stepwise sle=&sle;        /* modifiable SLENTRY */
run;

ods listing;
data _crit (keep=step AIC_: SC_:);          /* retrieve AIC
                                               and SC related */
set _crit(where =(Criterion in ('AIC','SC')));  /* values from
                                                   ODS data set   */
by step;
retain AIC_IO AIC_IC SC_IO SC_IC;
if first.step then do; AIC_IO=InterceptOnly;   /* regrouping
                                  variables for  customization*/

AIC_IC=InterceptAndCovariates;end;
if last.step then do;SC_IO=InterceptOnly;
SC_IC=InterceptAndCovariates;  output; end;
run;

data _crit;
merge _crit _summary;        /* merge with step-by-step data */
by step;                     /* for entered and  removed variables */
length Variable $100;
if EffectEntered ne ' '
then Variable=trim('+' || EffectEntered);  /* + in front of  entered
                                              variable */
else
Variable=trim('-' || EffectRemoved);        /* - in front of
                                              removed variable */
drop Effect: AIC_IO SC_IO;
run;

proc sql noprint;                    /* determine step where SC
                                        reaches minimum */
create table _sc as
select Step
from _crit
having SC_IC=min(SC_IC);
quit;

data _crit;                          /* put together tables with
                                        criteria */
retain
Step Variable Label DF NumberInModel
ScoreChiSq  WaldChiSq ProbChiSq  AIC_IC
diff_aic SC_ICC diff_sc;

merge _crit _sc(in=_2);
by Step;
retain aic_ret sc_ret;
if _2 then SC_ICC=put(SC_IC,11.2) || '*'; else      /* mark step with
                                                       minimum SC  */
           SC_ICC=put(SC_IC,11.2);
if _n_ =1 then do; aic_ret=aic_ic; sc_ret=sc_ic; end;
else
do;
diff_aic=aic_ret - aic_ic;       /* change in AIC at each step */
diff_sc=sc_ret - sc_ic;          /* change in SC at each step   */
aic_ret=aic_ic;
sc_ret = sc_ic;
end;
format diff_: 10.2;
drop aic_ret sc_ret label  ;
run;

ods listing close;
ods pdf file="select.pdf";                  /* output into PDF file */
ods proclabel="Stepwise Regression Table";

options ls=150;
                                            /* print the table */
proc print noobs width=min label;
var Step Variable DF NumberInModel
ScoreChiSq  WaldChiSq ProbChiSq
AIC_IC diff_aic SC_ICC diff_sc
;
label
    AIC_IC='AIC'
    SC_ICC ='SC'
    label = 'Label'
    ScoreChiSq='Score Chi- Square'
    WaldChiSq ='Wald Chi- Square'
    ProbChiSq ='Pr > Chi- Square'
     diff_aic='Decrease in AIC'
    diff_sc ='Decrease in SC'
    Variable='Variable Entered(+) or Removed(-)';

footnote  "NOTE: SLE=&sle IS USED FOR ENTRY INTO MODEL";
footnote2 '     * INDICATES MINIMUM OF SC          ';
run;

data _crit(keep=step value criterion);
set _crit;
criterion='AIC'; value=aic_ic; output;
criterion='SC '; value=sc_ic; output;
run;
                                        /* plot AIC and SC by step */

 goptions reset=global gunit=pct rotate=portrait
        cback=white colors=(black)
        htitle=2 htext=2.0 autofeed ftext="Helvetica/bold"
        device=pdf;

footnote1 justify=right "&sysdate &systime";
symbol1 interpol=join  value=dot  height=2 color=black;
symbol2 interpol=join  value=circle  height=2 color=black;

axis1 label=("Step") offset=(5,5)pct minor=none;
axis2 label=("Value")  offset = (5,5) pct;
ods proclabel="Plot of Criteria by Step" ;

 proc gplot data=_crit;
   plot
       Value*Step=criterion
      / haxis=axis1 vaxis=axis2;
run;

ods pdf close;

%mend;
```

3. Comparing Predictive Ability of Models

For comparing the predictive ability of logistic regression models, one useful measure is the area under the receiver operating characteristic (ROC) curve, abbreviated AUC. Previously, we presented a macro for calculating AUC

directly for weighted data from a survey, by summing the area of trapezoids (Izrael et al. 2002). As is shown in Hanley and McNeil (1982), an equivalent measure is the statistic *c*, which PROC LOGISTIC computes (for unweighted survey data) and displays in the Association of Predicted Probabilities and Observed Responses table; that is, the value of *c* equals AUC. Because *c* is usually more convenient computationally, we have also developed a macro for calculating it for weighted survey data.

The receiver operating characteristic curve is often used to describe the accuracy of tests in diagnostic medicine, as summarized in the review by Pepe (2000). Briefly, the test yields a numerical result X, such that larger values are more indicative of disease. One can choose a threshold z and dichotomize the test by defining $X \geq z$ as a positive result. From subjects whose true disease status is known (both diseased and nondiseased), one obtains the false-positive rate and the false-negative rate for each value of z. The ROC curve is obtained by plotting 1 minus the false-negative rate against the false-positive rate for all possible choices of z. That is, each value of z yields a point on the curve, which includes the point (0,0) (if z is high enough, the test produces no positives) and the point (1,1) (if z is low enough, all outcomes are positive).

The area under the ROC curve provides a summary of the accuracy of the diagnostic test. As Pepe points out, the AUC "can be interpreted as the probability that the test result from a randomly chosen diseased individual is more indicative of disease than that from a randomly chosen nondiseased individual." This interpretation or equivalence, discussed also by Hanley and McNeil (1982), focuses attention on the distributions of the test result (for example, the concentration of a chemical in blood) in diseased and nondiseased persons. If the two distributions are clearly separated, the probability will be close to 1; but if they are centered at the same value, the probability will be ½. In the context of logistic regression we refer to event cases and non-event cases, rather than diseased and nondiseased persons. The "test result" is the predicted probability of an event, from the logistic regression model.

The definition of *c* involves concordant and discordant pairs of observations. One begins by forming all pairs in which one case is an event and the other is a non-event. Denote the total number of pairs by *t*. A pair is *concordant* if the event observation has a higher predicted probability than the non-event observation; a pair is *discordant* if the event observation has a lower predicted probability than the non-event observation; and if the predicted probabilities are equal for the two observations, the pair is a tie. Denote the number of concordant pairs by *nc* and the number of discordant pairs by *nd*. Then the number of tied pairs is $t - nc - nd$. In unweighted data the formula for *c* is

$$c = (nc + 0.5(t - nc - nd))/t \, . \qquad (1)$$

Using SAS to estimate *c* for the weighted logistic regression

In this paper we investigate the behavior of *c* in a weighted logistic regression model. It turns out that in fitting a model, the Logistic procedure takes the survey weights into account, but it ignores them or uses them incorrectly in calculating the measures in the Association of Predicted Probabilities and Observed Responses table, the *c*-statistic in particular. One can test a model with a few predictors with and without weights to confirm that the Logistic procedure gives the same value of *c* for both of them. To calculate *c* in the presence of survey weights, we wrote a macro, *%WTC,* that takes the survey weights into account. We give an overview of the macro below and consider its application, both as a stand-alone program and as a subroutine in a bootstrap procedure.

The macro computes *c* based on the formula (1) calculating *weighted* components *nc, nd,* and *t.*

Let the number of event responses in a sample be *E* and the number of non-event responses be *N*. The total unweighted number of pairs being considered is *E*N*. Let us consider the (*i,j*) pair of observations, and let the weight and the predicted probability for the *event* observation (response is 1) be w_i and p_hat_i and for the *non-event* observation (response is 0) be w_j and p_hat_j. If p_hat_i is greater than p_hat_j, then, following the definition given above, the pair is *concordant* and its weighted representation $w_i * w_j$ is added to the weighted total of *concordant* pairs (*nc*).

Similarly, if p_hat_i is less than p_hat_j, then the pair is *discordant*, and its weighted representation $w_i * w_j$ is added to the weighted total of *discordant* pairs (*nd*). Finally, if the pair is neither *concordant* nor *discordant*, the product $w_i * w_j$ is added to the weighted total of *tied* pairs. Denoting W_E as the total *weighted* number of event responses and W_N as the total weighted number of non-event responses, the *weighted* total number of pairs is calculated as $t = W_E * W_N$. Based upon the weighted totals accumulated after *E*N* comparisons, the macro calculates the *c*-statistic by formula (1). The macro prints the value of *c* and stores it in a specified data set. Exhibit 2 presents the macro. We now describe its function section by section.

Ø contain the macro's input parameters; `model` represents the string of explanatory variables, all of which must be categorical (otherwise the number of distinct predicted probabilities could be very large); `depvar` is a response variable, assumed to have the value of 1 for an event and 0 for a non-event; `replica` must be blank when running the macro as a stand-alone program; otherwise it must be assigned the name of a macro variable that serves as a replicate counter when calculation of *c* is done for each bootstrap replicate.

Ù check that the variables in the model are present in the input data set. If not, the macro outputs names of absent variables into the LOG and stops execution.

Ú PROC LOGISTIC's statements and options. The data set `_PROBS` specified in the option `OUT` will include all variables of the input data set, along with the predicted probability `_P_HAT`.

Û calculate unweighted and weighted numbers that contribute to formula (1).

Ü calculate weighted *c* according to formula (1).

Exhibit 2. Macro *%WTC*

```
%macro wtc    ( ds    = ,      /* input data set   */          Ø
                outds  = ,    /* output data set */
                weight = ,   /* survey weight  */
                model =  ,   /* all explanatory variables must
                                        be categorical */
                depvar = ,   /* dependent variable */
                replica= ,     /* replicate number for
                                    bootstrap calculation  */
                        ) ;

%global control;                                                Ù
%let control = 1;
%macro check;
   %local dsid  i nullstr rc varnum;

%let model=%upcase(&model);
%let depvar=%upcase(&depvar);
%let weight=%upcase(&weight);
%let string=&model &depvar &weight;

%let i=1;
%let control=1;
%let dsid=%sysfunc(open(&ds));
%if &dsid ne 1 %then %do;
%put **** DATA SET &ds DOES NOT EXIST ***;
%let control =0;
%end;
%if &control=1 %then %do;
%do %until(%scan(&string,&i)=&nullstr);
  %let varnum=%sysfunc(varnum(&dsid,%scan(&string,&i)));
%if &varnum=0 %then %do;
   %let control=0;
   %put ;
   %put **** VARIABLE %scan(&string,&i) APPEARS IN THE
MODEL, BUT NOT IN THE INPUT DATA SET;
   %put ;
  %end;
  %let i=%eval(&i+1);
%end;

%let rc=%sysfunc(close(&dsid));

%put ;
%put ***** VARIABLE THAT CAN BE USED AS FLAG IS
CONTROL=&control *****;
%put ;
 %end;
%mend check;
 %if (&replica=) or (&replica=1) %then %check;
 %if &control = 0 %then %do;
  %put **** MACRO TERMINATED  BECAUSE OF ERRORS
ABOVE ******;
  %goto exit;
  %end;
   %else %do;

proc logistic descending
   noprint data=&ds;    /* 1 is event; 0 is non-event */          Ú
weight &weight./norm;   /* use normalized weight */
class &model;
model &depvar= &model;
output out=_probs(keep=&depvar &weight _p_hat)  predicted=_p_hat;
run;

proc sql noprint;                                                  Û
select sum(&weight) into: _tot_wgt   /* total weighted number
                                              of records */
from _probs;

select count(*) into: _tot_unw      /* total unweighted number
                                              of records */
from _probs;

select count(*) into: p1count      /* total unweighted number of
                                              events*/
from _probs
where &depvar =1;

select count(*) into: p0count   /* total unweighted number of non-
                                              events*/
from _probs
where &depvar =0;
quit;

data _probs;
set _probs;
&weight= &weight.*&_tot_unw./&_tot_wgt;   /*normalize
                                              weights */
run;

proc sql noprint;
select sum(&weight) into: _total1   /* total weighted number of
                                              events */
from _probs
where &depvar=1;

select sum(&weight) into: _total0   /* total weighed number of
                                              non-events */
from _probs
where &depvar=0;
quit;

   /* transpose predicted probabilities for events */                Ü

proc transpose data=_probs(keep=_p_hat &depvar where=(&depvar=1) )
          out=_probs1p(drop=_label_) prefix=ph1_;
var _p_hat;
run;

/* transpose weight for event records */

proc transpose data=_probs(keep= &weight &depvar
where=(&depvar=1))
          out=_probs1w(drop=_label_) prefix=w1_;
var &weight;
run;

          /* transpose predicted probabilities for non-event records */
proc transpose data=_probs(keep=_p_hat &depvar where=(&depvar=0))
out=_probs0p(drop=_label_) prefix=ph0_;
var _p_hat;
run;
```

```
                                /* transpose weights for non-event records */
proc transpose data=_probs(keep= &weight &depvar
where=(&depvar=0))
            out=_probs0w(drop=_label_) prefix=w0_;
var &weight;
run;

 /* accumulate weighted number of concordant and discordant pairs */
data out(keep= _discord _concord _tie);
retain _concord _discord _tie 0;

set _probs1p;
   if _n_=1 then set _probs1w;
    if _n_=1 then set _probs0p;
     if _n_=1 then set _probs0w;

                                        /* arrays with predicted probabilities */
 array p1[ &p1count] ph1_1 - ph1_%left(&p1count);
array p0[ &p0count] ph0_1 - ph0_%left(&p0count);
                                        /* arrays with weights */
array w1[ &p1count] w1_1 - w1_%left(&p1count);
array w0[ &p0count] w0_1 - w0_%left(&p0count);

do i=1 to &p1count;     /* accumulation of weighted discordant and*/
 do j=1 to &p0count;                                    /* concordant pairs */
     if p1[i]< p0[j] then _discord=_discord+w1[i]*w0[j];
else
     if p1[i]> p0[j] then _concord=_concord+w1[i]*w0[j];
else
     _tie=_tie+w1[i]*w0[j];
 end;
end;
run;

data &outds (keep=Wgt_c );   /* calculate weighted c by formula  (1)   */
set out;
total= &_total1 * &_total0;    /* total weighted number of pairs*/
Wgt_c=(_concord + .5*(total - _concord -_discord))/ total;
run;

proc print;
run;
%end;
 %exit:;
%mend;
```

4. The Bootstrap Method for Variance Estimation

The bootstrap (Efron 1982) uses resampling to provide a basis for studying the behavior of estimates. For a simple random sample of size *n*, with observations $x_1, x_2, ..., x_n$, the main steps involve setting *B* (the number of "bootstrap samples," usually large); using sampling with replacement to draw a bootstrap sample of *n*, $X_1^*, X_2^*, ..., X_n^*$, from the set $\{x_1, x_2, ..., x_n\}$ (*B* times, independently); and calculating the estimate, *t*, from each bootstrap sample to obtain $t_1^*, t_2^*, ..., t_B^*$. Analysis of the t_b^* then yields information on the sampling distribution of *t* when the data come from the population that underlies $x_1, ..., x_n$. For example, the sample standard deviation of the t_b^* is the bootstrap standard error of t.

As mentioned earlier, many analyses involve fitting two or more logistic regression models to the same survey data. Then, in choosing among the models, it may be useful to compare their predictive value, via the *c* statistic. To assess the size of the difference in the *c* statistic relative to the variation in the data (i.e., sampling variability), we need the estimated standard error of the difference. One suitable approach is the bootstrap method, which uses replication (Wolter 1985). The bootstrap involves drawing repeated independent samples (with replacement) from the original sample and then estimating the *c* statistic for each model and the difference in the *c* statistic, for each of these bootstrap samples. The sample standard deviation of that difference (over the bootstrap samples) is the bootstrap estimate of its standard error.

In practice, the survey data used to estimate logistic regression models almost never come from a simple random sample. Survey data generally come from complex sample designs that involve features such as stratification, multi-stage cluster sampling, unequal selection probabilities, and unequal sampling weights. Rust and Rao (1996) discuss the use of replication methods to obtain standard errors for complex survey designs. The application of the bootstrap procedure to a complex sample design involves drawing *B* bootstrap samples (replicates) within each stratum of the design. Application of the bootstrap requires that *B* be large. Efron and Tibshirani (1993) recommend $B = 200$. Korn and Graubard (1999) indicate that larger values of *B* (e.g., $B = 400$ or 800) yield more-stable variance estimators.

Rust and Rao (1996) give a method for adjusting the final sampling weights to obtain bootstrap weights. They also note, however, that for the variance estimators to remain close to unbiased, the weight adjustment steps applied to the original sample should be applied to each bootstrap replicate. This is likely to be an important consideration in most complex sample designs, given the considerable number of weight adjustments that are commonly made to obtain the final sampling weights. Thus, for each bootstrap replicate one should repeat all of the weight calculation steps. As a result each of the bootstrap replicates would have its own set of final sampling weights. Given the complexities of the bootstrap, it is wise to consult with a statistician who is familiar with the bootstrap method of variance estimation before creating bootstrap samples and bootstrap replicate weights.

Using SAS to create bootstrap replicates

Our data came from a stratified one-stage cluster sample of over 20,000 persons that incorporates several weighting adjustments. The sample design entails stratification of the U.S. into close to 100 geographic areas. The application of the bootstrap procedure to our sample design involves drawing the bootstrap samples (replicates) within each

stratum. In connection with other analyses of the same data, we had previously constructed 1,000 bootstrap replicates, in order to obtain bounds for 95% confidence intervals directly from the distribution of the bootstrap estimates, as well as bootstrap standard errors. Thus, it was natural to use those 1,000 replicates in estimating the standard error of the difference in *c*.

The following statements show how the SURVEYSELECT procedure was used to draw the 1,000 bootstrap replicates:

```
%let dd = ourdirectory;
%let ini_iter=1;
%let max_iter=1000;
%let in_smpfile=&dd..samplefile;
%let in_nsize= geo_area_tot;

 /* create a data set with sample sizes to be drawn from each stratum */

proc freq data=&in_smpfile;
 table geo_area/out=&in_nsize(rename=(count=_nsize_)
                              drop=percent);
run;

/* macro to draw 1000 bootstrap samples */

%MACRO BOOTREP;

%do i=&ini_iter %to &max_iter;

  proc printto new print="brep_&i..lst";
  proc printto new log="brep_&i..log";
  title3 " REPLICATE =&i URS selection";
  options pageno=1;

  proc surveyselect data=&in_smpfile
    method=urs
    sampsize=&in_nsize
    out=&dd..urs_&i  outhits;
    strata geo_area;
  run;

  proc freq data=&dd..urs_&i;
    tables NumberHits;
   run;
%end;
%MEND BOOTREP;
%BOOTREP
```

To draw the 1,000 sample replicates with equal probability and with replacement, we used `METHOD=URS` (Unrestricted Random Sampling). `SAMPSIZE = GEO_AREA_TOT` identifies the SAS data set that contains `_NSIZE_` , the different sample sizes for the strata. The `OUT=&dd..urs_&i` option outputs each of the 1,000 samples into a separate permanent SAS dataset. The `OUTHITS` option outputs a separate observation for each selection when an observation is selected more than once. The output dataset contains for each observation the variable `NumberHits`, the number of times a household was selected into the sample in a given replicate. The `STRATA` statement defines the variable `GEO_AREA` as the stratification variable.

Using SAS and the bootstrap replicates to estimate the variance of *c*

We demonstrate in Exhibit 3 the macro *%ALLREPL*, which uses the macro *%WTC* as a subroutine to refit a weighted logistic regression model and obtain *c* for each of 1,000 bootstrap replicates.

Exhibit 3: *%ALLREPL* Macro

```
%let youranal = analyt          /*analytic file with all data */
%let dsbswts = replwts;        /* data set with ID variable and replicate
                                                          weights */
%let model = yourmodel;    /* string with explanatory variables   */
%let depvar = yourresponse;  /*  response variable */

%macro allrepl (start,end);


%do v=&start %to &end ;        /* &START and &END are first and
                        last replicate to process, 1 and 1,000 by default */
data _analyt
merge &youranal (in=_1 )
&dsbswts (keep=ID w&v where=(w&v ne 0) in=_2);

  /* retrieve  &V-th replicate where  V-th replicate
                                        weight is not  zero */

by ID;
if _2;
wgt=w&v;
drop w&v;
run;

%wtc( ds = _ANALYT,           /*  calculate c for  &V-th replicate */
        outds  = C,
        id   = ID,
        weight = WGT,
        model =  &YOURMODEL,  /* refit model to data in &V-th
                                        replicate                    */
       depvar =  &YOURESPONSE,
       replica = &V,
       );
%end;
%mend;

%allrepl(1,1000)
```

5. Example

For our modeling we had a pool of nineteen potential explanatory variables (all of them categorical) `expl_var1 , … , expl_var19` and a two-level (1,0) response variable `present_or_absent`. Our goal was to build several weighted logistic regression models based on certain criteria, estimate their effectiveness by calculating the weighted *c*-statistic, and compare those models by assessing differences in *c* against the variation in the data applying the bootstrap approach. Data processing involved in our investigation is reflected in the flowchart in Figure A2 in the Appendix. The weighted logistic regression models used the stepwise selection in PROC LOGISTIC (SAS version 8.2). We normalized the weights so that the

sum of the weights equaled the unweighted number of cases.

In selecting the models we first satisfied our curiosity about how well we could do with the best single variable among 19 predictors – `expl_var1`. Using the macro *%WTC,* we calculated *c* for the one-variable model, which we denoted by *C1*. In selecting the full model, we first applied the macro *%SELECT* with SLENTRY=0.95, thus letting the maximum number of variables into the model. Table A1 and Figure A1 in the Appendix present the output of the macro *%SELECT*. According to Table 1A, the 14-variable model `(expl_var1, … , expl_var14)` was a good candidate for the full model. We confirmed this by obtaining the same set of predictors after applying the conventional stepwise model with the usual SAS default, SLENTRY=0.05.

Using the macro *%WTC,* we calculated *c* for the full model, which we denoted by *C14*. Analyzing Table A1 and Figure A1 in the Appendix, we selected a 6-variable model `(expl_var1, … , expl_var6)` as a reduced model because it gave the global minimum of SC (marked by * in the SC column of Table A1). We calculated weighted *c* for the reduced model using the macro *%WTC* and denoted it by *C6*.

The estimated predictive values of the three investigated models are shown in Table 1. They are: *C1* = 0.56965, *C6* = 0.64543, and *C14* = 0.66905. The addition of the 5 additional predictors increased *c* by a substantial amount, 0.076. On the other hand, adding in the next 8 predictors increased *c* by only 0.024, which is less than a third of the first increase.

To estimate the standard errors of assessed *c*'s and later of differences in *c*'s, we applied the bootstrap methodology to our data. Our sample design entails stratification of the U.S. into close to 100 geographic areas The application of the bootstrap procedure to our sample design involved drawing the bootstrap samples (replicates) within each stratum. In practice, we drew 1,000 bootstrap samples using the macro *%BOOTREP* described earlier.

Once we had drawn our bootstrap replicates, we had to run each of them through the entire process of weight calculation as described in Section 4. The process of calculating survey weights is a large undertaking and, therefore, is beyond the scope of this paper. We simply assume that we have 1,000 bootstrap replicates --- each with a set of survey weights calculated.

To calculate the standard errors of predicted values and of differences in predicted values, we first calculated *c*'s for each of the 1,000 replicates for each of three models using macro *%ALLREPL* described earlier. As a result, we obtained three sets of weighted replicate *c*'s: $CR1_i$, $CR6_i$, and $CR14_i$ (i = 1 to 1,000) for the one-variable, reduced, and full models, respectively.

To estimate the bootstrap standard errors of *C1*, *C6,* and *C14*, we simply applied PROC UNIVARIATE to $CR1_i$, $CR6_i$, and $CR14_i$ (i = 1 to 1,000) to obtain the standard deviations.

To estimate the standard error of the difference in *c* between the three models, $DIFF_{14,6} = C14 - C6$, and $DIFF_{6,1} = C6 - C1$, we applied PROC UNIVARIATE to the differences $CR14_i - CR6_i$ (i = 1 to 1,000) and $CR6_i - CR1_i$ (i = 1 to 1,000), respectively, to obtain the standard deviations $STD_{14,6}$ and $STD_{6,1}$. Then we calculate *t*-values using those differences and standard deviations:

$$t_{14,6} = \text{abs}(DIFF_{14,6} / STD_{14,6}) \text{ and}$$

$$t_{6,1} = \text{abs}(DIFF_{6,1} / STD_{6,1}).$$

The respective *p*-values are

$$p_{14,6} = (1\text{-PROBT}(t_{14,6}, \text{df}))*2 \text{ and}$$

$$p_{6,1} = (1\text{-PROBT}(t_{6,1}, \text{df}))*2$$

where df=1,000.

Calculated *c*-statistics, standard errors, *t*- and *p*- values are shown in Table 1.

Table 1. *c* Statistics and Bootstrap Standard Errors

Number of Predictors	*c* or Difference in *c*				
	c or Difference	Std. Err.	Variance	t	p
1	0.56965	0.00725	0.00005250		
6	0.64543	0.00716	0.00005130		
14	0.66905	0.00682	0.00004651		
6 vs 1	0.07578	0.00733	0.00005373	10.338	.0000
14 vs. 6	0.02362	0.00418	0.00001747	5.650	.0000

6. Discussion

In analyzing data from a survey, researchers often need to compare the effectiveness of several logistic regression models. We have used stepwise model selection in conjunction with the *%SELECT* SAS macro to calculate, tabulate, and plot the AIC and SC criteria at each step, and to determine the step at which SC reaches its minimum. The *c* statistic is an important measure of the predictive ability of a logistic regression model. Most survey data files have survey weights attached. The LOGISTIC procedure does not take the weights into account in its

calculation of *c* and therefore usually does not give the correct value. The SAS macro *%WTC* uses the survey weights in the calculation of *c*. We have used *c* to compare the predictive ability of three logistic regression models estimated from the same survey data file. Using bootstrap samples, it is possible to test the null hypothesis that the two models being compared have the same *c* value. We provide some background on how SURVEYSELECT can be used to create bootstrap samples and present the macro *%BOOTREP*, which implements the task.

References

Efron, B. (1982). *The Jackknife, the Bootstrap and Other Resampling Plans.* Philadelphia: Society for Industrial and Applied Mathematics.

Efron, B and Tibshirani, R.J. (1993*). An Introduction to the Bootstrap*. New York: Chapman and Hall.

Hanley, J.A. and McNeil, B.J. (1982). "The Meaning and Use of the Area under a Receiver Operating Characteristic (ROC) Curve," *Radiology*, 143, 29-36.

Izrael, D., Battaglia, A.A., Hoaglin, D.C., Battaglia, M.P. (2002). "Use of the ROC Curve and the Bootstrap in Comparing Weighted Logistic Regression Models," *Proceedings of Twenty-Seventh Annual SAS Users Group International Conference,* Paper 248.

Korn, E.L. and Graubard, B.I. (1999). *Analysis of Health Surveys*. New York, John Wiley & Sons, Inc.

Pepe, M.S. (2000). "Receiver Operating Characteristic Methodology," *Journal of the American Statistical Association*, 95, 308-311.

Rust, K.F. and Rao, J.N.K. (1996). "Variance Estimation for Complex Surveys Using Replication Techniques," *Statistical Methods in Medical Research*, 5, 283-310.

SAS Institute Inc. (1995). *Logistic Regression Examples Using the SAS System*, Version 6, First Edition. Cary, NC: SAS Institute Inc.

SAS Institute Inc. (1999). *SAS/STAT, Version 8*, Chapter 39. Cary, NC: SAS Institute Inc.

Shtatland, E.S., Moore, S., Dashevsky, I., Miroshnik, I., Cain, E., Barton, M.B. (2000) "How To Be a Bayesian in SAS: Model Selection Uncertainty in Proc Logistic and Proc Genmode," *Proceedings of North-Eastern SAS Users Group Conference, 2000,* pp. 724-732.

Wolter, K.M. (1985). *Introduction to Variance Estimation*. New York: Springer-Verlag.

Contact Information

David Izrael
Abt Associates Inc.
55 Wheeler St.
Cambridge, MA 02138
617-349-2434
david_izrael@abtassoc.com

Appendix

Table A1. AIC and SC and other statistics at each step of stepwise logistic regression model

Step	Variable Entered(+) or Removed(-)	Degrees of Freedom	Number In	Score Chi-Squared	Wald Chi-Squared	Pr > Chi-Squared	AIC	Decrease in AIC	SC	Decrease in SC
1	+EXPL_VAR1	2	1	269.5891	.	<.0001	22328.78	.	22352.85	.
2	+EXPL_VAR2	3	2	238.8323	.	<.0001	22092.00	236.78	22140.14	212.71
3	+ EXPL_VAR3	1	3	137.7295	.	<.0001	21955.92	136.08	22012.08	128.06
4	+ EXPL_VAR4	1	4	93.3166	.	<.0001	21863.21	92.71	21927.39	84.69
5	+ EXPL_VAR5	2	5	118.9826	.	<.0001	21748.10	115.12	21828.32	99.07
6	+ EXPL_VAR6	5	6	99.5123	.	<.0001	21661.57	86.53	21781.90*	46.42
7	+ EXPL_VAR7	50	7	197.4572	.	<.0001	21559.03	102.54	22080.47	-298.57
8	+ EXPL_VAR8	1	8	58.8086	.	<.0001	21503.37	55.66	22032.83	47.64
9	+ EXPL_VAR9	3	9	55.0349	.	<.0001	21455.39	47.98	22008.93	23.91
10	+ EXPL_VAR10	2	10	36.6041	.	<.0001	21423.72	31.68	21993.29	15.63
11	+ EXPL_VAR11	1	11	19.7777	.	<.0001	21406.19	17.53	21983.79	9.51
12	+ EXPL_VAR12	5	12	18.9453	.	0.0020	21397.52	8.67	22015.23	-31.44
13	+ EXPL_VAR13	2	13	10.2071	.	0.0061	21391.27	6.25	22025.03	-9.79
14	+ EXPL_VAR14	1	14	6.3577	.	0.0117	21387.03	4.25	22028.80	-3.78
15	+ EXPL_VAR15	1	15	2.2745	.	0.1315	21386.78	0.25	22036.57	-7.77
16	- EXPL_VAR15	1	14	.	2.2735	0.1316	21387.03	-0.25	22028.80	7.77

Figure A1: AIC and SC versus step of stepwise logistic regression model

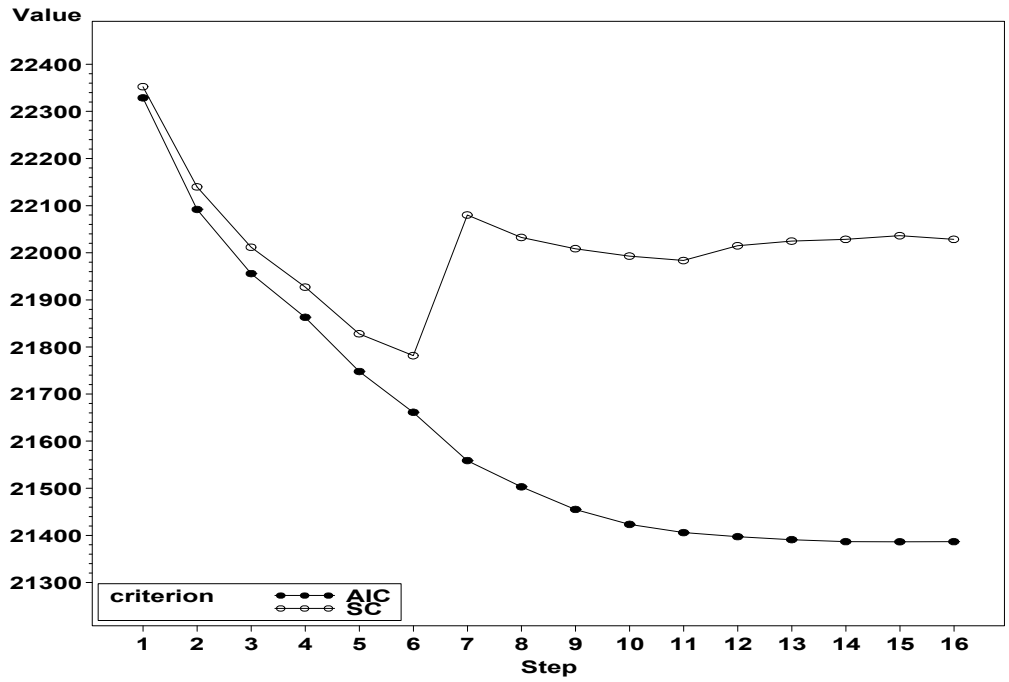

Figure A2 : Flowchart of data processing and analysis