MSUG Webinar November 20, 2025

Logistic Modeling without Split-Samples

Bruce Lund, Statistical Trainer

ABSTRACT

Here is a quote from: N. Kriegeskorte, et. al. (2009) "Circular analysis in systems neuroscience: the dangers of double dipping", *Nature Neuroscience* ⋯

> "Double Dipping is the use of the same dataset for selection and selective analysis. It gives distorted descriptive statistics and invalid statistical inference"

Double Dipping would arise if a logistic model was fit to an Analysis dataset and the same Analysis dataset was used for computing model validation statistics (e.g. c-statistic, average squared error, etc.).

The long standing approach to avoid double dipping is usage of split-sampling. In split-sampling the Analysis dataset is randomly divided into Training and Validation datasets. The split is often 50%-50% but could be 60%-40% or 70%-30%.

The focus of this paper is on binary logistic modeling. In split sampling, a logistic model is fit on Training, without ever looking at the Validation dataset. Once a final model is fitted, then model performance is measured on the Validation dataset.

Can the problem of double dipping be avoided without a split-sample? If so, this would have the advantage of fitting the model on the entire Analysis dataset, giving better predictor variable selection and better coefficient estimation. But this leaves open the question of how to perform model validation.

It is a purpose of the paper to show how split-sampling can be avoided. Briefly, this approach involves the usage of bootstrap sampling to find the “optimism correction”. This “optimism correction” is an adjustment to performance metrics (e.g., c-statistic, average squared error, etc.) that are computed on the full Analysis dataset. That is, the model is fitted on the Analysis dataset. The performance metrics are also computed on the Analysis dataset, but these performance metrics are then corrected by an “optimism correction”. The paper explains how bootstrap sampling is utilized in finding the optimism correction. Optimism Correction is presented in a book by Efron and Tibshirani (1993), *An Introduction to the Bootstrap*, pp 247-252. In recent years, F. Harrell and E. Steyerberg (and others) have championed this approach (see references in paper).

THE ELEMENTS OF A LIFT CHART

A discussion is needed to describe a performance metric called the “Lift Chart”. A Lift Chart is computed for a binary-target model. In this paper, this model will be a logistic model. It is assumed a population of prospects has been identified for eligibility for a target marketing campaign, and a random sample from the population has been taken. In the early sections of this paper the model fitting and validation discussion will be based on the split-sample approach composed of a Training dataset and a Validation dataset.

A Lift Chart is computed for the Training sample once a model has been fitted. Examination of this Lift Chart is part of the process in determining whether this model is ready for the validation process.

Then, the model’s performance on the Lift Chart from the Validation sample will determine the success of the model (in conjunction with other performance metrics).

Criteria for assessing the performance of the model on a Lift Chart are given in a later section.

SAS CODE TO CREATE A LIFT CHART:

This SAS code prepares the data, fits a logistic model to Training, and creates a Lift Chart for this model on the Training dataset.

A dataset WORK is created with the target variable Y having exactly 50% events and 50% non-events. An even number of events and non-events is often achieved by the modeler via preliminary oversampling of events. But the merits and features of oversampling is not a topic for this paper.

Here is dataset WORK. The two predictors, X1 and X2, provide a weak relationship to the target Y.

```
%LET SEED = 3;
%LET N = 1000;
options missing=' ';
DATA WORK;
call streaminit(&SEED);
DO ID = 1 to &N;
    if mod(ID,2) = 0 then Y = 1;
    else Y = 0;
    if Y = 1 then do;
          X1 = rand('normal') + .2;
          X2 = rand('normal') + .2;
          end;
    if Y = 0 then do;
          X1 = rand('normal') - .2;
          X2 = rand('normal') - .2;
          end;
    output;
    end;
PROC FREQ DATA = WORK; table Y;
TITLE "DATASET WORK: SEED=&SEED, N=&N";
run;
```

SAS Code 1: Create dataset WORK with 50% events

It is assumed that the reader is familiar with PROC SURVEYSELECT. Here, SURVEYSELECT is used to create TRAIN and VALDATE datasets with each having 500 observations, and, importantly, with each having an event-rate of 50%. The SAS comments below briefly describe the syntax.

```
/* Must SORT by STRATA variable Y before PROC SURVEYSELECT */
/* Since OUTALL is specified, variable SELECTED is created by SURVEYSELECT
This causes ALL observations to be output to WORK */
/* SELECTED=1 means this observation was assigned to VALIDATE */
/* SELECTED=0 means this observation was assigned to TRAIN */
/* n = (250, 250) says: sample 250 0's and 250 1's to assign to VALIDATE */
/* SRS = simple random sampling */
/* 500 observations remain. 250 are events. These are assigned to TRAIN */
PROC SORT DATA = WORK; BY Y;
PROC SURVEYSELECT DATA = WORK SEED = 1
OUT = WORK(rename=(SELECTED=PART)) OUTALL /* Note: SELECTED is now PART */
METHOD = SRS /* sampling without replacement */
N = (250 250);
STRATA Y;
run;
```

SAS Code 2: PART specifies Train and Validate, each with 500 observations and 50% event-rate

A modeler might choose to forego the use of PROC SURVEYSELECT and, instead, assign observations to Train and Validate by random “coin flips”.

```
If rand('uniform') < 0.5 then PART = 0; /* This is Train */
  Else PART = 1; /* This is Validation */
```

This approach should never be taken because the event-rates on Train and Validate will not be equal. This inequality of event rates will invalidate a performance metric called “calibration” for the Validation Lift Chart. An illustration of this problem is given in a section which follows. Lift Chart performance metrics are also discussed in a section which follows. Now back to the SAS code for the example.

PROC HPLOGISTIC creates the logistic model using dataset Train. The Partition statement informs HPLOGISTIC that observations with PART = 0 are Training observations and observations with PART = 1

are Validation. The Copyvars statement is needed to specify which variables are output to dataset SCORE1. These variables are Y and PART. The P (the probability from the model) is also output because P=P appears in the OUTPUT OUT statement.

```
PROC HPLOGISTIC DATA = WORK;
PARTITION rolevar = PART(train="0", validate="1");
MODEL Y (descending) = X1 X2;
OUTPUT OUT=SCORED1 COPYVARS=(Y PART) P=P;
run;
```

SAS Code 3: PROC HPLOGISTIC Fits a Model to Train and Scores both Train And Validate

Suppose the HPLOGISTIC model is called MODEL1. Dataset SCORE1 has probabilities, the P's, fitted by MODEL1 for all observations in WORK.

The WHERE statement in PROC RANK reads in only the Training observations. PROC RANK will rank the P's from SCORED1 that are found in the Training dataset.

```
PROC RANK DATA = SCORED1(WHERE=(PART=0)) out = RANK1 descending
GROUPS = 10;
VAR P;
RANKS Rank_P;
run;
```

SAS Code 4: Train is ranked by P, ten ranks are created, Output to RANK1

Crucially, the modeler has chosen to create 10 ranks via the GROUPS = 10 statement. Ten ranks (deciles), are often used, by tradition, for a Lift Chart. But this number of ranks may be inappropriate.

The number of ranks should not exceed the limit imposed by properties of the Lift Chart called "calibration" or "monotonicity". More about "calibration" and "monotonicity in a section below.

The rank variable is named Rank_P. Traditionally, (as is done here), the rank with the highest P's is numbered as 0. The Rank_P's are numbered 0, 1, 2, ….

PROC MEANS creates some of the statistics for the Lift Chart including the average P by rank (P_mean), the 99th P percentile and the 1st P percentile for each rank (P99 and P01), the event-rate (Y_mean) by rank, and _Freq_ which is the number in each rank. In addition, these statistics are computed for all observations in RANK1. The output dataset is MEANOUT1.

```
PROC MEANS DATA = RANK1 noprint; class Rank_P; var Y P;
output out = MEANOUT1 (drop = _type_)
P1(P) = P01
P99(P) = P99
MEAN(P) = P_mean
MEAN(Y) = Y_mean;
run;
```

SAS Code 5: One output observation for ALL of RANK1 and one observation for each rank

In the SAS code below there is a macro variable &Z_value=1.28. At modeler's discretion, this value could be changed.

The DATA Step reads MEANOUT1 and creates _Diff_ , _Lift_, and P_In_Out for each output observation.

Diff For ALL and each Rank_P, _Diff_ = Y_mean – P_mean
Lift: For each Rank_P: _Lift_ = Y_mean (Rank_P) / Y_mean (ALL) … expressed as a percentage.
P_In_Out: For each Rank_P: P_In_Out has value either "IN" or "OUT" as determined by the rule below:
If P_mean lies within the interval below, then "IN". Else "OUT".
-&Z_value * SD + Y_mean, +&Z_value * SD + Y_mean … (*)
Where SD = sqrt(Y_mean * (1 – Y_mean) / Freq)

Statistic "SD" is patterned after the standard deviation from flipping a coin "Freq" times where the probability of heads is Y_mean. But the underlying Y's in Y_mean are constrained to be associated with a

P which falls within the range of the particular rank. Interval (*) is a pseudo 80% confidence interval for the “true” Y_mean for a rank. This true Y_mean is estimated by P_mean. P_In_Out measures whether P_mean falls in the pseudo 80% confidence interval.

```
%LET Z_value = 1.28;
DATA MEANOUT1; set MEANOUT1;
retain Y_mean_ALL; drop Y_mean_ALL;
if _N_ = 1 then do;
    Y_mean_ALL = Y_mean;
    end;
_Diff_ = Y_mean - P_mean;
if _N_ > 1 then do;
    _Lift_ = Y_mean / Y_mean_ALL;
    Low = -&Z_value*sqrt(Y_mean*(1-Y_mean)/_freq_) + Y_mean;
    High = &Z_value*sqrt(Y_mean*(1-Y_mean)/_freq_) + Y_mean;
    If Low < P_mean < High then P_In_Out = "IN ";
          else P_In_Out = "OUT";
    end;
rename _freq_ = Freq;
run;
```

SAS Code 6: Create _Diff_ , _Lift_, and P_In_Out for the Lift Chart

Finally, **PROC PRINT** prints the Lift Chart for the Training dataset.

```
PROC FORMAT;
value ALL . = "ALL";
PROC FORMAT;
value LIFT . = "";
PROC PRINT DATA = MEANOUT1 NOOBS;
VAR Rank_P Freq P01 P99 P_mean Y_mean _Diff_ _Lift_ Low High;
var P_In_Out / style(data)={just=c};
format P01 P99 P_mean Y_mean _Diff_ Low High 5.3;
format Rank_P ALL.;
format _Lift_ percent6.1;
TITLE "TRAINING, SEED=&SEED Z_value=& Z_value";
run;
```

SAS Code 7: PROC PRINT of Lift Chart for Train

Here is the printout from PROC PRINT.

Rank_P	Freq	P01	P99	P_mean	Y_mean	_Diff_	_Lift_	Low	High	P_In_Out
ALL	500	0.27	0.733	0.5	0.5	0				
0	50	0.637	0.765	0.675	0.68	0.005	136%	0.596	0.764	IN
1	50	0.589	0.637	0.609	0.54	-0.069	108%	0.450	0.630	IN
2	50	0.551	0.589	0.568	0.58	0.012	116%	0.491	0.669	IN
3	50	0.528	0.551	0.539	0.64	0.101	128%	0.553	0.727	OUT
4	50	0.502	0.528	0.513	0.50	-0.013	100%	0.409	0.591	IN
5	50	0.480	0.502	0.489	0.54	0.051	108%	0.450	0.630	IN
6	50	0.450	0.48	0.464	0.48	0.016	96%	0.390	0.570	IN
7	50	0.411	0.449	0.432	0.32	-0.112	64%	0.236	0.404	OUT
8	50	0.364	0.411	0.388	0.36	-0.028	72%	0.273	0.447	IN
9	50	0.211	0.364	0.323	0.36	0.037	72%	0.273	0.447	IN

Figure 1: Lift Chart using 10 ranks. P_In_Out flags when P_mean is IN or OUT of the Interval:
Low = -&Z_value*sqrt(Y_mean*(**1**-Y_mean)/Freq) + Y_mean;
High = &Z_value*sqrt(Y_mean*(**1**-Y_mean)/Freq) + Y_mean;

PERFORMANCE CRITERIA OF A LIFT CHART

CALIBRATION: Calibration is agreement between Y_mean and P_mean, by rank, as measured by P_In_Out. The Lift Chart of Figure 1 fails calibration for Rank_P's 3 and 7. But note: _Diff_ = 0 for ALL. This is forced to be true by nature of the maximum log-likelihood equation solution.

MONOTONICITY looks at the column _Lift_ in Figure 1 to assess whether _Lift_ values are monotonically decreasing. Although generally decreasing, these _Lift_ values fail to be monotonic.

A "Consistent" Lift Chart satisfies **calibration** (P_In_Out = "IN") and has **monotonicity** down the ranks

LIFT: A performance metric is "LIFT". LIFT looks at the column for _Lift_ at Rank_P = 0 and _Lift_ at the lowest rank (in this case, Rank_P = 9). But measurement of LIFT in Figure 1 is distorted by the lack of monotonicity. The lowest _Lift_ occurs at RANK_P = 7.

In targeted marketing, Model1 may be a "propensity to buy model". In this case, the prospects in the top _Lift_ are often chosen for a marketing promotion. If, instead, Model1 is a "loyalty-defection model" where the model probability gives the likelihood that the next purchase by the customer will be a loyal purchase, then the bottom rank indicates extreme defection and the top rank indicates extreme loyalty. Either rank might be targeted by a campaign (with distinctive messaging).

The LIFT in Figure 1 shows there is, at least, some potential benefit provided by Model1 but the usage of 10 ranks is *inconsistent* with respect to calibration and monotonicity.

WHAT TO DO WITH THIS LIFT CHART?

One approach is to "patch up" by the Chart by ad hoc combining of ranks. This approach may create ranks of unequal size and, by selective combining of ranks, may make the model look "too good".

An alternative is to use the following heuristic formula for the ideal number of ranks (see Appendix A for a derivation). The formula specifies using K ranks where:

$$K = (P\_Range / (2 * Z\_value * SD))^{2/3}$$

P_Range = P99 – P01 where P99 and P01 appear in the ALL row (see below).
Z_value = 1.65 where 1.65 is chosen to provide an added margin to satisfy P_In_Out requirements of the pseudo confidence interval for Y_mean … where &Z_value had been set to 1.28.
SD = Sqrt(Y_mean * (1-Y_mean) / Freq) … where Y_mean and Freq appear in the ALL row (see below). Here, Freq is the sample size for Train.

Here are statistics for ALL for Train:

Rank_P	Freq	P01	P99	P_mean	Y_mean
ALL	500	0.270	0.733	0.5	0.5

In this example: K = (0.463 / (2 * Z_value * 0.022361)) $^{2/3}$ = 3.40 … rounding up to 4.

Using `GROUPS = 4` the Lift Chart is consistent with respect to calibration and monotonicity.

Rank_P	Freq	P01	P99	P_mean	Y_mean	_Diff_	_Lift_	Low	High	P_In_Out
ALL	500	0.270	0.733	0.500	0.500	-.000				
0	125	0.566	0.754	0.629	0.616	-.013	123%	0.545	0.687	IN
1	125	0.502	0.566	0.532	0.560	0.028	112%	0.487	0.633	IN
2	125	0.435	0.501	0.470	0.480	0.010	96%	0.407	0.553	IN
3	125	0.224	0.433	0.369	0.344	-.025	69%	0.274	0.414	IN

Figure 2

However, the K from the formula is *not guaranteed* to produce a Training Lift Chart which is consistent or which will find the maximum number of ranks for a Lift Chart which is consistent.

Another alternative is to rerun PROC RANK with GROUPS = 9, 8, …, until a consistent Lift Chart is found.. It turns out that `GROUPS = 6` is the maximum number of ranks that give a consistent Lift Chart.

Rank_P	Freq	P01	P99	P_mean	Y_mean	_Diff_	_Lift_	Low	High	P_In_Out
ALL	500	0.270	0.733	0.500	0.500	-.000				
0	83	0.601	0.765	0.652	0.627	-.025	125%	0.539	0.714	IN
1	83	0.543	0.600	0.569	0.578	0.009	116%	0.489	0.667	IN
2	84	0.502	0.542	0.522	0.560	0.038	112%	0.471	0.648	IN
3	83	0.458	0.502	0.482	0.518	0.036	104%	0.428	0.608	IN
4	84	0.396	0.456	0.430	0.381	-.049	76%	0.294	0.468	IN
5	83	0.211	0.395	0.346	0.337	-.008	67%	0.252	0.422	IN

Figure 3: Six is the Maximum number of Ranks that give a consistent Lift Chart for Train

But the _Lift_ column for `GROUPS = 6` is not substantially superior to the _Lift_ for `GROUPS = 4`.

A prudent choice might be 4 ranks. Here, 125 prospects have lift 123% while only 83 prospects from the 6 rank approach have lift 125%

The _Lift_ of 123% for the top rank, as given by `GROUPS = 4`, might appear to be low. But _Lift_ must be viewed in the context of the profitability of incremental purchases as well as the characteristics of the population. [1, 2] A _Lift_ of 123% could be profitable for the company

PERFORMANCE METRICS FOR MODEL1 ON VALIDATION DATASET

The Lift Chart (and other performance statistics such as c-statistic, misclassification rate, etc.) help the modeler decide if MODEL1 is ready for validation. When ready, the Lift Chart and other performance metrics, computed on Validation, will determine whether or not the model meets its prescribed goals.

The target marketing company may have a standard for the number of ranks for a Validation Lift Chart. In this situation, the number of ranks is pre-specified. For example, company management might routinely expect to see LIFT results for a top *quartile* or *decile*.

Otherwise, a maximum number of ranks for the Validation Lift Chart would be the highest number of ranks that yielded a consistent Lift Chart. But how to find such a number of ranks?

The modeler might determine this number of ranks for Validation by using the maximum number of ranks on Train that yielded a consistent Lift Chart.

In the current example this was 6 ranks. But this number of ranks could fail to be consistent for a Validation Lift Chart due to the independent random character of the Validation sample.

Instead, a modeler might use this heuristic formula for the number of ranks, K, for the Validation Lift Chart.

$$K = (P\_Range / (2 * Z\_value * SD))^{2/3}$$

To avoid “peeking” at the Validation dataset, the inputs for this formula would taken from Training, with the exception of “Freq” which is the size of the Validation dataset. A default for Z_value is 1.65.

To create a Lift Chart for Validation, the only needed change to the SAS code from the code for the Training Lift Chart is shown below (i.e., “1” replaces “0”):

```
PROC RANK DATA = SCORED1(where=(PART=1)
```

[1] The notion of incrementality requires that a control group be randomly sampled from a prospect list. This allows a measurement of incrementality after the running of the marketing campaign. This topic is not discussed in this paper.
[2] It must be noted that Lift also depends on the choice of Population. A wider choice, with many unlikely responders, will meaninglessly increase the Lift of Rank_P=0. The target marketing company must bear this in mind.

Here is the Validation Lift Chart with 4 ranks.

Rank_P	Freq	P01	P99	P_mean	Y_mean	_Diff_	_Lift_	Low	High	P_In_Out
ALL	500	0.267	0.736	0.497	0.500	0.003				
0	125	0.563	0.767	0.626	0.624	-.002	125%	0.553	0.695	IN
1	125	0.497	0.561	0.527	0.552	0.025	110%	0.479	0.625	IN
2	125	0.433	0.495	0.464	0.504	0.040	101%	0.431	0.577	IN
3	125	0.234	0.431	0.373	0.320	-.053	64%	0.252	0.388	IN

Figure 4

Commentary:

- P_mean, at 0.497, is very near to the Y_mean of 0.5, so there is overall consistency.
- All values of P_In_Out are “IN”.
- The _Lift_ column is monotonic.
- The _Lift_ values very similar to that from the Lift Chart on Train.

Everything looks good! But, maybe not !! More about this “maybe not” in a later section.

First, here is a discussion of how *not* to split the sample into Train and Validation and why not.

USING RAND(‘UNIFORM’) TO SPLIT SAMPLE.

Earlier in the paper it was mentioned that Train and Validation should not be assigned by the simple random process shown below:

```
%LET SEED = 6;
DATA WORK; SET WORK;
call streaminit(&SEED);
if rand('uniform') < 0.5 then PART = 0;
    else PART = 1;
```

SAS Code 8: “Coin Flip” approach to assigning observations to Train and Validate

If this code had been applied, then the Lift Chart for Training is given here.

Rank_P	Freq	P01	P99	P_mean	Y_mean	_Diff_	_Lift_	Low	High	P_In_Out
ALL	488	0.271	0.712	0.488	0.488	0				
0	122	0.550	0.727	0.607	0.590	-0.016	121%	0.533	0.647	IN
1	122	0.486	0.549	0.515	0.549	0.034	113%	0.492	0.607	IN
2	122	0.430	0.486	0.459	0.459	0.001	94%	0.401	0.517	IN
3	122	0.246	0.428	0.370	0.352	-0.018	72%	0.297	0.408	IN

Figure 5A

Lift Chart for Validation is given below.

Rank_P	Freq	P01	P99	P_mean	Y_mean	_Diff_	_Lift_	Low	High	P_In_Out
ALL	512	0.271	0.733	0.496	0.512	0.016				
0	128	0.562	0.747	0.623	0.633	0.010	124%	0.578	0.687	IN
1	128	0.496	0.560	0.528	0.570	0.042	111%	0.514	0.626	IN
2	128	0.429	0.495	0.464	0.508	0.044	99%	0.451	0.564	IN
3	128	0.230	0.427	0.369	0.336	-0.033	66%	0.283	0.389	IN

Figure 5B

Overall, Validation Lift Chart looks good. But Y_mean for Training is 0.488 and for Validation it is 0.512. This creates this problem: The Lift Chart for Validation might be used as a forecast model by the target marketing company. Suppose company draws a new sample from the population and scores these

prospects using Model1. Using the Lift Chart for Validation, the company might forecast a 63.3% response rate for the top rank. But this forecast is overstated because it is based on a Validation sample with Y_mean = 0.512 while Model1 was fitted on Train with Y_mean = 0.488.

Of course, the company could patch-up the Lift Chart by multiplying the Y_mean's by an adjustment factor = 0.488 / 0.512. But this is messy, and would be confusing when explained to management.

For N = 1000, the _Diff_ for ALL is 0.016. If N is increased to 5000, then _Diff _ = 0.014 for ALL (using the Seed = 6 as before). At N = 10000, then _Diff _ = 0.010. But, regardless of the value of N, there is never a good reason to skip the use of PROC SURVEYSELECT.

VALIDATION USES A RANDOM SAMPLE, SO WHAT COULD GO WRONG?

Now it's time to explain the "but, maybe not" comment made earlier. Could a particular random assignment of Train and Validation, by chance, give substantially different results when computing the Lift Chart performance metric on Validation?

This question is explored by the exercise below:

In PROC SURVEYSELECT the stratified sampling selection depends on the SEED statement.

```
PROC SORT DATA = WORK; BY Y;
PROC SURVEYSELECT DATA = WORK
OUT = WORK(rename=(SELECTED=PART)) OUTALL
METHOD = SRS /* sampling without replacement */
N = (250 250)
SEED = 1;
STRATA Y;
run;
```

SAS Code 9. The SEED statement in PROC SURVEYSELECT determines the random split

Leaving all other code the same, Lift Charts were computed for Seeds = 1, …, 20. The logistic model from SEED = 1 is the model that has been discussed so far. The other 19 splits show how the vagaries of random chance can sometimes lead to unsatisfactory modeling results.

The results are summarized in Figure 6 below. A "N" (= No) is an unsatisfactory outcome.

	Train			Validation		
Seed	Calibration	Monotonic	Lift: Hi / Low	Calibration	Monotonic	Lift: Hi / Low
1	Y	Y	1.23 / 0.69	Y	Y	1.25 / 0.64
2	Y	Y	1.30 / 0.66	N	Y	1.18 / 0.67
3	Y	Y	1.18 / 0.66	Y	Y	1.25 / 0.66
4	N	N	1.14 / 0.70	N	Y	1.28 / 0.66
5	Y	Y	1.22 / 0.64	Y	Y	1.22 / 0.67
6	Y	Y	1.33 / 0.74	N	N	1.10 / 0.59
7	Y	Y	1.18 / 0.69	Y	Y	1.25 / 0.67
8	Y	Y	1.17 / 0.66	Y	Y	1.30 / 0.66
9	N	N	1.15 / 0.69	N	N	1.26 / 0.64
10	Y	Y	1.28 / 0.58	Y	Y	1.23 / 0.77
11	Y	Y	1.22 / 0.74	N	Y	1.26 / 0.64
12	Y	Y	1.26 / 0.64	Y	Y	1.23 / 0.70
13	Y	N	1.09 / 0.70	N	Y	1.34 / 0.62
14	Y	Y	1.28 / 0.66	N	N	1.14 / 0.66
15	N	N	1.20 / 0.66	Y	Y	1.20 / 0.69
16	Y	Y	1.38 / 0.69	N	N	1.14 / 0.69
17	Y	Y	1.36 / 0.59	N	N	1.07 / 0.77
18	Y	Y	1.30 / 0.66	N	Y	1.17 / 0.67
19	Y	Y	1.26 / 0.62	N	Y	1.18 / 0.70
20	Y	Y	1.31 / 0.61	N	Y	1.17 / 0.74

Figure 6. Results of 20 random splits to Training and Validation datasets

For seeds 4, 9, 13, 15 the modeler might abandon the effort to fit a model, concluding that relationship between the target and predictors was too weak to a fit. Otherwise, the modeler might conclude that the promising model failed to validate for samples with seed 6, 14, 16, 17. By random assignment of splits to Train and Validation, these four models: Model6, Model14, Model16, Model17 would be withdrawn, despite having acceptable Lift Charts on Train. Bad luck! It turned out that Model1 is something of an anomaly.

REFLECTIONS AND A PROPOSAL FOR HOW TO PROCEED

To be sure, this is a cautionary tale for the usage of split-sampling when N ~ 1000 and event-rate ~ 0.5.

Now, N = 1000 with event-rate = 0.5 is fairly small when considered as the Analysis dataset for a target marketing campaign model. What if N = 20,000 with event-rate = 0.5? It is probable that Validation results would not depend much on a particular split. Split sampling for such a dataset is sound. Yet, in the final section of this paper, the argument is made that split-sampling can be avoided regardless of the size of N. The replacement method is called "Optimism Correction". Optimism Correction is introduced after a brief discussion of bootstrap sampling.

BOOTSTRAP SAMPING

Before proceeding, a quick discussion is given of how to perform bootstrap sampling. Suppose dataset BOOT has 5 observations … BOOT = {0, 1, 2, 3, 4}.

A bootstrap sample from BOOT is formed by 5 random picks from BOOT with replacement. One bootstrap sample is {0, 0, 1, 3, 4}. Note: "0" has been sampled twice since sampling is with replacement.

PROC SURVEYSELECT can perform bootstrap sampling. Here, two bootstrap samples are selected:

```
DATA BOOT;
DO X = 0 to 4;
  OUTPUT;
  END;
PROC SURVEYSELECT DATA=BOOT OUT=BootSamples NOPRINT SEED=111
METHOD=URS /* with replacement */
SAMPRATE=1 /* Sample = 100% */ REPS=2; /* Create 2 Samples */
PROC PRINT DATA=BootSamples;
run;
```

SAS Code 10

By sheer chance, the first bootstrap sample is {0, 1, 2, 3, 4} !!
The second bootstrap sample is {0, 2, 3, 3, 3}. This second sample has 3 "hits" for X = 3.

Obs	Replicate	X	NumberHits
1	1	0	1
2	1	1	1
3	1	2	1
4	1	3	1
5	1	4	1
6	2	0	1
7	2	2	1
8	2	3	3

Figure 7

OPTIMISM CORRECTION

Would Model1 be much better if fit to all 1000 observations of WORK versus N = 500 (keeping seed = 1)? In Figure 7 below the performance statistics on their respective Train datasets are shown.

The Model on the Analysis dataset where N = 1000 is somewhat superior. Also, notably, the coefficient for X2 is rather different when fit to N = 1000. This would reflect a better estimate (since using more data).

	Training	
Statistic	N=1000	N=500
Area under the ROCC (c-stat)	0.6241	0.6197
Average Square Error	0.2385	0.2397
Misclassification Error	0.417	0.414
Mean Difference	0.04597	0.04101

Parameter	Estimate	Estimate
Intercept	0.01386	0.00665
X1	0.3361	0.3364
X2	0.2911	0.2447

Figure 8

But if all the Analysis data are used, then how can Model1 be validated? The approach of bootstrap sampling applied to optimism correction provides an answer.[3] Below, the focus will be on the Lift Chart.

Here are the steps to obtain the optimism correction for the Lift Chart when all N = 1000 are used to fit the model.

1. Decide on the number of ranks. Summary results of Model1 are shown below. Since this report is created for the purpose of determining the number of ranks and not in measuring performance, this is not a “double dipping”.

Rank_P	Freq	P01	P99	P_mean	Y_mean
ALL	1000	0.252	0.751	0.5	0.5

   The formula for estimating the number of ranks gives K = 4.51. Might use 4 or maybe 5.

$$K = (0.499 / (2 * 1.65 * 0.01581))^{2/3} = 4.51$$

   The Lift Chart with K = 5 did not satisfy calibration. Instead, the Lift Chart is created with 4 ranks. This Lift Chart will be called $LC_{apparent}$ (following the established practice for naming).

2. From the Analysis database, 200 (typically) bootstrap samples are created. Here is the code.

```
PROC SURVEYSELECT DATA=WORK OUT=BootSamples
NOPRINT SEED=111 METHOD=URS /* with replacement */
SAMPRATE=1 /* Sample = 100% */ REPS=200; /* Create 200 Samples */
```

   For each of the 200 samples, a logistic model is fit to the sample with target Y and predictors X1, X2. The models and the Lift Charts are saved. The Lift Charts will be called LC_{boot} (without regard to which sample is being considered).

3. For each bootstrap model (from step 2), the WORK dataset is scored by this model and the Lift Charts are saved. There will be 200 Lift Charts. These will be called LC_{full}.

4. For each of the 200 bootstrap samples, LC_{boot} - LC_{full} is computed. The subtraction symbol means the entries for each rank of LC_{full} are subtracted from the corresponding entries from each rank of LC_{boot}.

5. The optimism for the Lift Chart is $LC_{optimism}$ = average(LC_{boot} - LC_{full}), i.e., averaged over the 200 bootstrap samples.

6. Now, the optimism corrected Lift Chart is $LC_{corrected}$ = $LC_{apparent}$ - $LC_{optimism}$. This is the Lift Chart that should be reported for the Lift Chart performance of Model1 on WORK.

[3] Validation based on cross-validation is another approach, This is not discussed in this paper.

This process might seem like “smoke and mirrors”. But optimism correction was developed by the renowned statistician Bradley Efron, and it is well established.[4] Frank Harrell and Ewout Steyerberg (among others) are leaders in promoting this approach.[5] Below, it is applied to WORK and Model1.

APPLY OPTIMISM CORRECTION TO LIFT CHART FOR MODEL1 ON WORK N=1000.

The Lift Chart, **$LC_{apparent}$**, is shown below:

Rank_P	Freq	P01	P99	P_mean	Y_mean	_Diff_	_Lift_	Low	High	P_In_Out
ALL	1000	0.252	0.751	0.500	0.500	-.000				
0	250	0.573	0.773	0.637	0.604	-.033	121%	0.564	0.644	IN
1	250	0.500	0.572	0.534	0.568	0.034	114%	0.528	0.608	IN
2	250	0.430	0.499	0.465	0.496	0.031	99%	0.456	0.536	IN
3	250	0.209	0.428	0.363	0.332	-.031	66%	0.294	0.370	IN

Figure 9. The Apparent Model

Average(LC_{boot}), the average Lift Chart of 200 bootstrap samples, is shown on the left. **Average(LC_{full})**, the average Lift Chart of the 200 bootstrap models applied to WORK, is shown on the right.

Bootstrap Models		
RANKP	P_mean	Y_mean
ALL	0.49966	0.49966
0	0.63901	0.61595
1	0.53505	0.56305
2	0.46421	0.48807
3	0.36058	0.33179

Scored Bootstrap on Full Data		
RANKP	P_mean	Y_mean
ALL	0.49960	0.5
0	0.63947	0.61238
1	0.53475	0.56174
2	0.46382	0.48652
3	0.36036	0.33936

Figure 10A. 200 bootstrap samples

The **Optimism** is Boot minus Full. **Optimism Corrected** is **$LC_{apparent}$ - Optimism**

Optimism = Boot - Full		
RANKP	P_mean	Y_mean
ALL	0.00006	-0.00034
0	-0.00046	0.00357
1	0.0003	0.00131
2	0.00039	0.00155
3	0.00022	-0.00757

Optimism Corrected = Apparent - Optimism						
RANKP	P_mean	Y_mean	SD(Y)	%Y +/- 1.28*SD(Y)		IN/OUT
ALL	0.49994	0.50034	0.031	0.561	0.640	P IN
0	0.63746	0.60043	0.031	0.527	0.607	P IN
1	0.53370	0.56669	0.032	0.454	0.535	P IN
2	0.46461	0.49445	0.030	0.301	0.378	P IN
3	0.36278	0.33957	0.031	0.561	0.640	P IN

Figure 10B. 200 bootstrap samples

The modeler should report the Lift Chart of Figure 10B on the right (Optimism Corrected) as the “official” Lift Chart performance metric for Model1 as fit to the WORK.

In the same manner the optimism correction of the c-statistic and average squared error were computed.

Statistic	Apparent Model	Avg. Bootstrap Model	Avg. Full Dataset (*)	Optimism	Corrected Optimism
c-statistic	0.6241	0.6264	0.6229	0.0035	0.6206
average squared error	0.2385	0.2377	0.2393	-0.0016	0.2401

(*) Full Dataset is Scored by Bootstrap Models

Figure 11. 200 bootstrap samples

[4] See: Efron and Tibshirani (1993) *An Introduction to the Bootstrap*, pp 247-252 (but difficult to read)

[5] See: Harrell, F. (2015) *Regression Modeling Strategies: With Applications to Linear Models, Logistic Regression, and Survival Analysis, 2nd Ed*, and Steyerberg, E. (2019). *Clinical Prediction Models 2nd Ed*.

SAS CODE FOR OPTIMISM CORRECTION

- SAS code for Figures 10 and 11 is given in Appendix B. Some additional features were commented out but can be restored by uncommenting. More discussion is given in Appendix B.
- A SAS Global Forum paper gives an example of Optimism Correction and a SAS Macro. See "Estimating Harrell's Optimism on Predictive Indices Using Bootstrap Samples" by I. Stijacic Cenzer, Y. Miao, K. Kirby, W. J. Boscardin (2013).
- The recent book *Building Regression Models with SAS*®, by Robert Rodriguez (2023) includes a discussion of optimism correction for the logistic model (pp. 172-174). SAS code is provided via the authors website.[6] This code computes the optimism correction for the misclassification rate.

THE CHALLENGE OF "PURE" OPTIMISM CORRECTION

A modeler might have a continuous numeric predictor X and may wonder whether or not to include X^2. The modeler might plot X versus Y (e.g. averaging X and Y across small intervals of X) to look for a quadratic shape.[7] This, of course, involves looking at target Y.

If this process was performed on the Analysis dataset, instead of the split-sample Train dataset, then a subsequent optimism correction calculation would be compromised by a double dipping.

A similar discussion would apply to binning discrete predictor Z using algorithms that involve Y.[8]

What to do? There is a way out, although this way out, involves additional (perhaps complex) programming. Here are the steps, at a high level:

1. PREPARE predictors in the Analysis dataset. This might involve transforming continuous predictor X or binning discrete Z's. Next, fit the model to the Analysis dataset using a variable SELECTION method.
2. Create 200 (or some large number) of bootstrap samples.
3. Perform the same predictor variable PREPARATION on each bootstrap sample (as in step 1). Now fit the model to the bootstrap samples using the same variable SELECTION method as was used when fitting the model to the Analysis dataset.
4. Proceed, as before, by scoring the Analysis dataset for each bootstrap sample
5. As before, compute optimism and optimism correction.

Example: The question of including X^2 in the model can be handled by variable selection methods.

For example, consider `SELECTION=BACKWARD`, as shown below. Here, X and X*X are both included in the variable list. The `HIERARCHY` statement (`single` is the default) will not remove X if X*X is still in the model.

```
PROC LOGISTIC DATA = Analysis desc;
CLASS C1 C2;
MODEL Y = X X*X X1-X5 C1-C2 / SELECTION=BACKWARD SLS=.05 HIERARCHY=single;
run;
```

When fitting models for the bootstrap datasets the same `SELECTION` statement is included in this **`PROC LOGISTIC`**. This handles the issue of X^2 without a double dipping. Below is the code for fitting the models on the bootstrap samples. Note: `FREQ` is included in **`PROC LOGISTIC`** since the bootstrap samples include a variable `NumberHits` which count the number of times an observation is sampled. The

[6] https://support.sas.com/downloads/package.htm?pid=2688
[7] A systematic approach is Function Selection Procedure. See SAS code in Lund, B. (2018) "The Function Selection Procedure" SAS Global Forum https://www.sas.com/content/dam/SAS/support/en/sas-global-forum-proceedings/2018/2390-2018.pdf
[8] Lund, B. (2024) "Binning Predictors for Logistic Regression", MSUG Conference https://www.misug.org/uploads/8/1/9/1/8191072/blund_binning_predictors.pdf

bootstrap sample is summarized by `NumberHits`. (Refer to Figure 8, replicate 2 for an example.) The 200 bootstrap sample are processed one at a time. See `(where=(replicate=&I))`.[9]

```
PROC LOGISTIC DATA=BootSamples(where=(replicate=&I)) desc OUTMODEL=OM;
FREQ NumberHits;
CLASS C1 C2;
MODEL Y= X X*X X1-X5 C1-C2 / SELECTION=BACKWARD SLS=.05 HIERARCHY=single;
```

Each of the 200 models is saved by `OUTMODEL=OM` and is then used to score the Analysis dataset.

With this code, usage of X^2 is handled without double dipping. See Appendix B for more commentary.

More Comments:

- An ambitious method for transforming continuous numeric predictors X is to use regression splines. Regression splines are compatible with the optimism correction approach. See Appendix D for an example.
- Regarding binning discrete predictors, I have not seen a binning method that fits neatly into the bootstrap model building step (step 3 above). An alternative is to simply make subjective decisions about binning Z, without looking at Y, and implement in the Analysis dataset.

Last Comments:

The variable selection step (e.g. `SELECTION=BACKWARD`) is relatively straight forward to integrate into the optimism correction process.

But what about "data exploration" that involves looking at X's and Y? This is a very normal pre-modeling process which gives the modeler a feel for the data and for potential modeling approaches. Is this double dipping? In the words of Kriegeskorte, et. al. (2009), is this "the use of the same dataset for selection and selective analysis"? I think it is.

EVENTS PER PREDICTOR, EPP

Let n1 be the number of events in the Analysis dataset and let n0 = number of non-events so that N = n1 + n0. Commonly, n1 ≤ n0.

Let M = number of predictors, measured in terms of degrees of freedom, that are candidates for a logistic model. "Candidates" refers to predictors considered for the model, not to those ultimately selected.

Events per predictor (EPP) is simply EPP = n1 / K. For example, if n1 = 500 and M = 25, then EPP = 500/25 = 20. A question of considerable research interest is:

What is an acceptable minimum for EPP ?

An often cited rule is n1 / K > EPP = 10, "At least Ten Events per Predictor". But EPP guidelines are an area of active research. See the paper by Van Smeden, et. al. (2017). "No rationale for 1 variable per 10 events criterion for binary logistic regression analysis".[10]

Harrell [p. 72] [11] suggests EPP = 15. Steyerberg [p. 55] cites studies that support EPP = 10 or 20.[12]

This is a complex subject with no consensus, much less, simple rules. But it is maintained by all researchers that the higher the EPP, then the more reliable and stable would be the modeling results.

---

[9] BY REPLICATE processing is not used since the MODEL content is being saved in OUTMODEL=OM. Another programmer might find a way to use BY processing (to achieve efficiency and still save the models to OM).
[10] Van Smeden, et. al. (2017) "No rationale for 1 variable per 10 events criterion for binary logistic regression analysis", *BMC Medical Research Methodology*.
[11] Harrell, F. (2015) *Regression Modeling Strategies: With Applications to Linear Models, Logistic Regression, and Survival Analysis, 2nd Ed*
[12] Steyerberg, E. (2019). *Clinical Prediction Models* 2nd Ed.

EPP considerations may place limits on split-sampling decisions. Consider an Analysis dataset with N = 1000, event-rate = 0.5, and M = 30. Assume Train will have event-rate = 0.5. Then Train must have N ≥ 600 to satisfy the 10 EPP rule. But now the Validation dataset may be too small?

OPTIMISM CORRECTION: EMPIRICIAL EVIDENCE THAT IT WORKS

This section summarizes an illuminating study by Austin and Steyerberg, hereafter referred to as A-S.[13]

THE STUDY BY AUSTIN AND STEYERBERG (2017)

A-S used data from the EFFECT Study (Enhanced Feedback for Effective Cardiac Treatment) with 16,237 patients for their study of optimism correction versus split-sampling as applied to logistic modeling with a binary response. These 16,237 patients constituted the Master file. The response variable had an overall event-rate of 0.319. The model performance measure that was studied by A-S was the c-statistic.

The Study Design: Samples with EPP levels of 5 to 100 in increments of 5 (5, 10, 15, …, 100) were taken according to the following sampling design: (This required a massive amount of computing.)

> For each EPP level, simple random samples from the 16,237 Master File (replacing each sample after selection) were taken until there were 1000 samples ⋯ these 1000 samples were selected to have the targeted EPP (e.g. 5, 10, etc.), so many samples would be rejected, not having the targeted EPP.

Logistic models were fit to these samples. There were ten model predictors (M = 10), and they were always included in each model. Logistic modeling in this study involved only the fitting of the coefficients. There was no discussion of pre-modeling predictor preparation nor predictor variable selection methods.

Average c-statistics (across the 1000 samples) were computed for each EPP level for the four scenarios shown below. In the discussion below, "current sample" will refer to one of these samples.

(1) **A_c_stat**: Logistic model (the "Apparent model") is fit to a current sample and c-statistic is computed for this model. **A_c_stat** is the average across the 1000 samples for a given level of EPP.

(2) **S_c_stat**: Current sample is split 50%. Logistic model is fit to Train and c-statistic is computed on Validation. **S_c_stat** is the average across the 1000 samples for a given level of EPP.

(3) **O_c_stat**: 100 bootstrap samples were taken from a current sample, logistic models were fit and c-statistics computed. Bootstrap models then scored the current sample and c-statistics were computed. Optimism is computed as average bootstrap c-statistic minus average current sample c-statistic. Optimism Corrected is c-statistic from Apparent model c-statistic (see 1) minus Optimism. **O_c_stat** is average Optimism Corrected c-statistic across the 1000 samples for a given level of EPP.

(4) **E_c_stat**: Apparent Model is scored on all 16,237 patients minus the dataset from (1) and the c-statistic is computed. **E_c_stat** is the average across the 1000 samples for a given level of EPP.

Remarks about the current sample sizes (where M = 10 = number of predictors):

Consider (1): if EPP=5, then 5=n1 / M so that n1 = 50. Size of the current sample is N = 50/0.319 = 157.

Consider (2): if EPP=5, then Train has N = 157/2 = 78.

Consider (4): if EPP=5, then current sample has N = 16,080 = 16,237 - 57.

Fact: The c-statistic from fitting the 10 predictor model to the full 16,237 Master file = 0.741

For any dataset (call it WORK) the Apparent model is fit to WORK. The c-statistic for the Apparent model, when computed on WORK, is biased upward. But if the Apparent model is used to score the "Population", then this c-statistic would be the "true" c-statistic for the Apparent model.

E_c_stat (4) is the closest to "true". Of course, in practice, this E_c_stat is unknowable, as there would be no Master File. The other three c-statistics: A_c_stat, S_c_stat, and O_c_stat approximate E_c_stat with

[13] Austin, P. and Steyerberg, E. (2017). Events per variable (EPV) and the relative performance of different strategies for estimating the out-of-sample validity of logistic regression models, *Stat Methods Med Res.*

various levels of success. These approximations are made for the various levels of EPP. Now it is time to discuss these three approximations.

KEY FIGURE FROM AUSTIN AND STEYERBERG

Figure 1 of the paper by A-S appears on page 802 of *Statistical Methods in Medical Research* 2017, Vol. 26(2).

It is reproduced below (with my annotations) as Figure 12.

Here are several conclusions:

- A_c_stat (Apparent) is upward biased. That is, it is higher than E_c_stat (External) but converges as EPP increases.
- S_c_stat (Split-Sample) is downward biased. That is, it is lower than E_c_stat but converges as EPP increases.
- O_c_stat (Bootstrap corrected optimism) is essentially equal E_c_stat for EPP of 20 and beyond, and much closer to E_c_stat than is S_c_stat, at all EPP.
- S_c_stat at 2*EPP is comparable to O_c_stat at EPP. See, for example, S_c_stat at EPP = 40 (box A) and O_c_stat at EPP = 20 (box B).

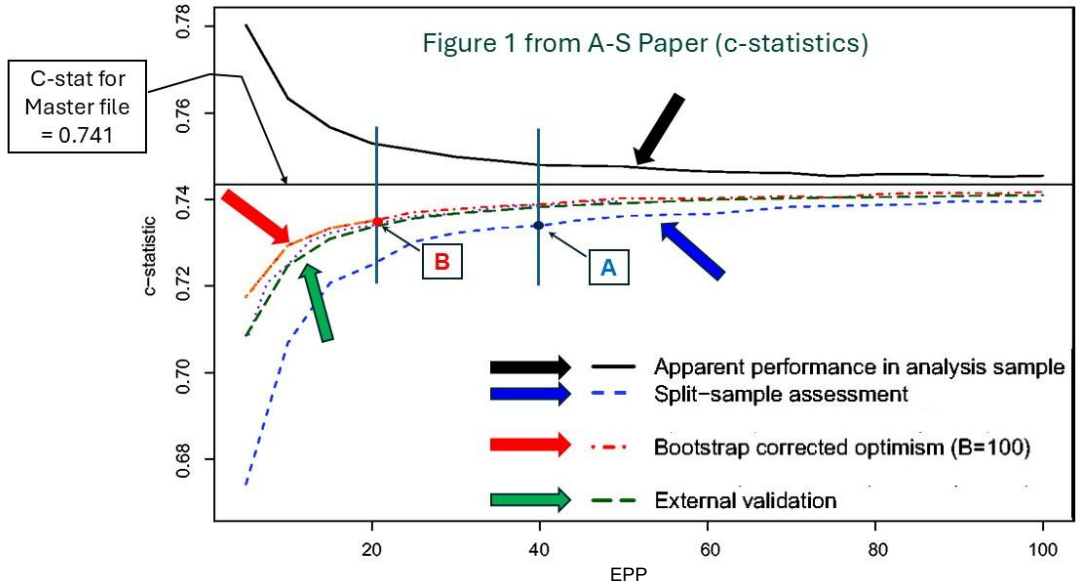

Figure 12. Adapted from Figure 1 of the paper by Austin and Steyerberg

A striking result found by A-S is that 50-50 split-sampling requires twice the dataset size versus optimism correction to reach approximately the same c-statistic.

Of course, these results above pertain to averages over 1000 iterations at each EPP. Any given iteration could provide a wide deviation from the averages.

MY STUDY WITH SIMULATED DATA

Motivated by A-S, I performed a simulation study to replicate some findings of the A-S paper.

1. METHODOLOGY FOR THE SIMULATION … GENERATION OF DATA

Analysis datasets were generated in this simulation which have N observations. The values of N are 1000, 2000, 3000, 4000, 5000, 6000. In addition, an additional 22,000 observations as an "external" sample were generated. A total random sample consists of N + 22,000 observations.

For each level of N there are 50 independent instances of a total sample. That is, N + 22,000 observations are randomly generated 50 times. All together, as N goes from 1000 … 6000 in increments of 1000, there are 6*50 = **300** instances. For each instance of N + 22,000 observations, there is a target variable Y (levels 0, 1) and 9 predictors (see code below for definition of the target and predictors).

The SAS code for this simulation study is given in Appendix C. Some of this code is reproduced below. This code is part of a macro. Two macro parameters appear, `&Sample` and `&Seed`. The user designates `&Sample` as 1000, 2000, …, or 6000. The user also assigns a random `&Seed`.

There is an Analysis dataset called `SAMPLE` and an External dataset called `POPULATION` that has 22,000 observations. Both `SAMPLE` and `POPULATION` are recreated for each instance.

Predictors `X1-X7 C12 C34` will appear in logistic models in the simulation, where `C12 C34` are class variables, each with 3 levels. Therefore, the predictors have 11 d.f., i.e. M = 11.

```
DATA SAMPLE POPULATION;
do ID = 1 to &Sample + 22000;
PART = MOD(ID,2);
ALL = 0; /* used later */
EXTERNAL = (ID > &Sample); /* Identifies the 22,000 */
cumLogit = ranuni(&Seed);
e = 1*log( cumLogit/(1-cumLogit ));
X1 = rannor(&Seed);
X2 = rannor(&Seed);
X3 = rannor(&Seed);
X4 = rannor(&Seed);
X5 = rannor(&Seed);
X6 = rannor(&Seed);
X7 = X6*X5;
B1 = (ranuni(&Seed) < .4);
B2 = (ranuni(&Seed) < .6);
B3 = (ranuni(&Seed) < .5);
B4 = (ranuni(&Seed) < .5);
C12 = B1 + B2;
C34 = B3 + B4;
Z = -6 + X1**2 + log(X2+8) + .01*X3 + 2*X7 + 0.1*B1 + B2 + B3 + B4 + 2*e;
Y = (Z > 5.0);
IF ID <= &SAMPLE THEN OUTPUT SAMPLE;
OUTPUT POPULATION;
end;
run;
```

SAS Code 11

The c-statistic for Apparent model is computed on the External dataset with N = 22,000. This is analogous to (4) from the A-S study. The 22,000 observations in Validation are those where variable `EXTERNAL` = 1 where `EXTERNAL = (ID > &Sample);`

`PROC HPLOGISTIC` fits a model to Train, having one of 1000, …, 6000 observations, and with the other 22,000 as the Validation dataset as specified by Partition variable `EXTERNAL`.

```
ODS OUTPUT PartFitStats = PartFitStats(rename=(Validation = E_c_Stat));
PROC HPLOGISTIC DATA= POPULATION;
PARTITION ROLE=EXTERNAL (TRAIN="0", VALIDATE="1");
CLASS C12 C34;
MODEL Y (descending) = X1-X7 C12 C34;
SELECTION METHOD=NONE;
run;
```

SAS Code 12

Run times for the simulation for each N (where N: 1000, …, 6000) are exceedingly long. A user may need to breakup the simulation runs into smaller chunks and then recombine. See comments in Appendix C.

2. MORE COMMENTS ON THE SET-UP

For each of the 50 samples of an N (where N = 1000, …, 6000), the event rate (%Y=1) varies randomly. The events per predictor (EPP) for a sample is given by: EPP = (number of events in sample) / 11.

Here is a table showing the average EPP for each N:

N	EPP (averaged over 50 samples)
1000	6.88
2000	13.82
3000	20.60
4000	27.50
5000	34.54
6000	41.16

Figure 13. Average EPP across the 50 samples for each level of N = 1000, …, 6000

Four modeling processes are applied to each of the 6*50 = 300 samples, These are numbered: (1), (2), (3), (4) and are described below. SAS code below is “pseudo code” to describe the c-statistics.

(1) The Apparent Model: The Apparent Model is the logistic model shown below.

PROC LOGISTIC DATA = Sample_of_N; CLASS C12 C34; MODEL Y = X1 - X7 C12 C34;

The c-statistic is computed. All together there are 50 c-statistics for each N and 300 in total. The average c-statistic for a given N is called A_c_stat.

(2) The Split-Sample Model: Each sample of N is split 50-50 into Train and Validate. The logistic model is fit on Train. The c-statistic is computed on Validation.

```
PROC LOGISTIC DATA = Train; MODEL Y = X;
SCORE DATA = Validate FITSTAT;
```

All together there are 50 c-statistics for each N and 300 in total. The average c-statistic for a given N is called S_c_stat.

(3) Bootstrap Sampling with Optimism Correction: For each sample: 50 bootstrap samples are taken. (Harrell recommends 200 bootstrap samples but due to computing resources, I reduced this number of bootstrap samples to 50 ⋯ this 50 is *different* from my earlier use of 50!)

The Logistic Model for each bootstrap sample:

```
PROC LOGISTIC DATA =  BootSample_of_Sample_of_N OUTMODEL = OM;
CLASS C12 C34; MODEL Y = X1 - X7 C12 C34;
```

The next step is to score the Bootstrap logistic model on the Analysis database and do this 50 times.

Finally, optimism corrected c-statistic is computed. The average c-statistic for a given N is called O_c_stat

(4) External: The External c-statistic comes from scoring the Apparent Model on the sample of 22,000.

```
PROC LOGISTIC DATA = Sample_of_N; CLASS C12 C34; MODEL Y = X1 - X7 C12 C34;
SCORE DATA = Sample_of_22000 FITSTAT;
```

The c-statistic is computed from the scores on the Sample_of_22000. All together there are 50 c-statistics for each N and 300 in total. The average c-statistic for a given N is called E_c_stat.

E_c_stat is the best approximation to the “true” c-statistic for the Apparent model and this “true” c-statistic is what the modeler wants to report. Of course, E_c_stat is unknowable in practice. The other three

c-statistics: A_c_stat, S_c_stat, and O_c_stat are compared to E_c_stat. This comparison is made for each of the sample sizes: N = 1000 to 6000.

Which one gives the best approximation to the “true” c-statistic?

3. RESULTS

Here is a Table that summarizes the results of the simulation study.

	Average c-statistic for 50 samples for each N					
E_c_stat	0.716	0.728	0.731	0.733	0.734	0.735
A_c_stat	0.762	0.745	0.744	0.744	0.743	0.742
S_c_stat	0.700	0.712	0.720	0.724	0.728	0.729
O_c_stat	0.724	0.724	0.730	0.732	0.734	0.735
N	1000	2000	3000	4000	5000	6000
Average EPP	6.88	13.82	20.60	27.50	34.54	41.16

Figure 14. Average c-statistics for each level of N for the four modeling processes.

Summary:

- For EPP at 13.82 (N=2000) the average O_c_stat was only 0.004 below the E_c_stat.
- At EPP of 20.60 (N=3000) and higher, the average c-statistics for E and O were essentially equal.
- Average A_c_stat, is consistently too optimistic. At EPP of 20.60 the over-estimate was 0.013 vs. E_c_stat.
- Average c-stat for Split Sampling is consistently too pessimistic. At EPP of 20.60 the under-estimate was 0.011 vs. E_c_stat.
- S_c_stat at EPP = 27.50 is the same as O_c_stat at EPP = 13.82. Likewise, S_c_stat at EPP = 41.16 is about the same as the O_c_stat at EPP=20.60. This is consistent with the finding by Austin and Steyerberg that says that c-statistic of split-sample at 2*EPP is roughly equal to the optimism corrected c-statistic at 1*EPP.

In conclusion: In A-S Study, 50-50 split-sampling required about twice the dataset size versus optimism correction to reach approximately the same c-statistic. Furthermore, the optimism correction c-statistic, O_c_stat, is much closer to the external c-statistic, E_c_stat, at all EPP.

IS THIS THE END OF SPLIT-SAMPLING FOR LOGISTIC MODELS?

I think the answer is YES for samples where EPP is low, perhaps under 20. For low EPP, Optimism Correction allows a model to be fit to the full Analysis dataset. This allows more predictors to enter the model and with better estimation of coefficients. But the modeler must try to include all predictor preparation steps within the Optimism Correction process. This may be challenging.

I think the answer is sort-of YES for samples with large EPP and few X’s. There is no harm in using Optimism Correction, except for added programming and added computational requirements.

But there is a downside to Optimism Correction for large EPP and many X’s. If the modeler is not already a subject-matter expert, then a split-sample allows the modeler to explore the relationship between the predictors and the target in a free flowing manner and, as well, to try various modeling techniques, while still preserving the Validation sample to avoid double dipping.

MSUG November 20, 2025, webinar

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author, Bruce Lund, at:
blundz@iCloud.com or blund.data@gmail.com.

APPENDIX A

This Appendix gives a heuristic formula for the number of ranks to be used in the Lift Chart for a logistic model. First, definitions are given:

Y_mean =	The event-rate for the dataset.
Freq =	Number of observations in the dataset.
SD =	Sqrt(Y_mean*(1-Y_mean)/Freq)
Z_value =	1.65 where 1.65 is chosen for the pseudo 90% confidence interval (CI) shown below for the "true" population Y_mean: CI = (-1.65 * SD + Y_mean, +1.65 * SD + Y_mean)
Freq_rank =	Number of observations within a rank (it is assumed that Freq_rank applies to all ranks).
SD_rank =	Sqrt(Y_mean*(1-Y_mean)/Freq_rank)
P_range =	P99 – P01 where P99 and P01 where P99 is the 99th percentile of P for the fitted model and P01 is the 1st percentile of P for the fitted model. (These percentiles are across ALL observations.)
K =	Number of ranks for the Lift Chart (to be solved for)

Here is the formula for K:

$$K = (P\_range / (2 * Z\_value * SD))^{2/3}$$

Derivation

This constraint determines K. The K summed lengths of the 90% CI's must equal P_range.

$$P\_range = K * 2 * Z\_value * SD\_rank \ldots (A)$$

An identity for K is given by:

$$Freq\_rank = Freq / K \ldots (B) \ldots \text{(note: Freq\_rank is unknown):}$$

Substituting the expression for Freq_rank from (B) into (A) gives:

$$P\_range = K * 2 * Z\_value * sqrt[Y\_mean*(1-Y\_mean) * K / Freq]$$

$$P\_range = K * 2 * Z\_value * SD * sqrt(K)$$

Solving for K:

$$K = (P\_range / (2 * Z\_value * SD))^{2/3}$$

Examples:

If: P_range = 0.8	Z_value = 1.65	Y_mean = 0.5	Freq = 5000,	then K = 10.55
If: P_range = 0.8	Z_value = 1.65	Y_mean = 0.48	Freq = 500,	then K = 4.90
If: P_range = 0.65	Z_value = 1.65	Y_mean = 0.1	Freq = 10000,	then K = 16.27
If: P_range = 0.99	Z_value = 1.65	Y_mean = 0.1	Freq = 10000,	then K = 21.54

But the formula is not guaranteed to give a consistent Lift Chart or, otherwise, may not give the maximum number of ranks that give a consistent Lift Chart. But it gets the user "into the ballpark" for finding a good value of K.

APPENDIX B

Here is the SAS code to compute optimism correction for MODEL1 fitted to WORK (N = 1000).

```
%LET SEED = 3;
%LET N = 1000;
options missing=' ';
DATA WORK;
call streaminit(&SEED);
DO ID = 1 to &N;
     if mod(ID,2) = 0 then Y = 1;
     else Y = 0;
     if Y = 1 then do;
             X1 = rand('normal') + .2;
             X2 = rand('normal') + .2;
             end;
     if Y = 0 then do;
             X1 = rand('normal') - .2;
             X2 = rand('normal') - .2;
             end;
     output;
     end;
```

SAS Code 13

Code to create FIGURES 11 and 12.

```
%LET DATASET = WORK;
%LET TARGET = Y;
%LET C_VARS = ;
%LET X_VARS = X1 X2;
%LET SELECTION = BACKWARD;
%LET SLS = .9999; /* Set high to disable SELECTION */
%LET SLE = .9999; /* That is, Keep all X's */
%LET GROUPS = 4;
PROC SURVEYSELECT DATA= &DATASET
OUT=BootSamples
NOPRINT SEED=111
METHOD=URS /* Sample with replacement */
SAMPRATE=1 /* Sample rate 100% */
REPS=200; /* Number of boot strap samples */
run;
%MACRO REP(R);
/* Clean-up Datasets that appear in PROC APPEND */
%IF %SYSFUNC(EXIST(BASE1)) = 1 %THEN %DO;
PROC DELETE DATA = BASE1;
run;
%END;
%IF %SYSFUNC(EXIST(BASE2)) = 1 %THEN %DO;
PROC DELETE DATA = BASE2;
run;
%END;
%IF %SYSFUNC(EXIST(BASE3)) = 1 %THEN %DO;
PROC DELETE DATA = BASE3;
run;
%END;
%IF %SYSFUNC(EXIST(BASE4)) = 1 %THEN %DO;
PROC DELETE DATA = BASE4;
run;
%END;
%IF %SYSFUNC(EXIST(BASE5)) = 1 %THEN %DO;
PROC DELETE DATA = BASE5;
run;
%END;
%IF %SYSFUNC(EXIST(ScoreFitStat1)) = 1 %THEN %DO;
PROC DELETE DATA = ScoreFitStat1;
run;
%END;
%IF %SYSFUNC(EXIST(ScoreFitStat2)) = 1 %THEN %DO;
PROC DELETE DATA = ScoreFitStat2;
run;
```

```
%END;
%IF %SYSFUNC(EXIST(NumberinModel)) = 1 %THEN %DO;
PROC DELETE DATA = NumberinModel;
run;
%END;
%IF %SYSFUNC(EXIST(Lift_Chart1)) = 1 %THEN %DO;
PROC DELETE DATA = Lift_Chart1;
run;
%END;
%IF %SYSFUNC(EXIST(Lift_Chart2)) = 1 %THEN %DO;
PROC DELETE DATA = Lift_Chart2;
run;
%END;
/* Fit Logistic Model to Bootstrap Samples … no BY … instead need to have OUTMODEL */
%DO I = 1 %TO &R;
ods exclude all;
/* Save MODEL information using OUTMODEL = OM */
PROC LOGISTIC DATA = BootSamples(where=(replicate=&I))
desc OUTMODEL = OM;
FREQ NumberHits;
CLASS &C_VARS;
MODEL &TARGET = &X_VARS &C_VARS
/ SELECTION=&SELECTION SLE=&SLE ;
SCORE DATA = BootSamples(where=(replicate=&I)) FITSTAT
OUT=SCORED1(keep = Replicate &TARGET P_1 NumberHits);
ods output ScoreFitStat = ScoreFitStat1;
ods output ConvergenceStatus = ConvergenceStatus;
ods output ModelBuildingSummary=ModelBuildingSummary;
run;
* Create Lift Charts for Boot Sample Models;
PROC RANK DATA= SCORED1
OUT= RANKOUT1
GROUPS=&GROUPS DESCENDING;
VAR P_1; /* variable that is ranked */
RANKS RANKP; /* name of ranks */
PROC MEANS DATA= RANKOUT1 NOPRINT;
FREQ NumberHits;
CLASS RANKP; VAR P_1 &TARGET Replicate;
OUTPUT OUT= Lift_Chart1 MEAN= PREDICT_B Y_B Replicate;
run;
/* END Create Lift Charts for Bootstrap Sample Models; */
/* For information: Record number of predictors in each Boot Strap Model */
/* If SELECTION is active, then count number of variables in model */
/*
DATA NumberinModel(keep=NumberinModel); Set ModelBuildingSummary end=eof;
if eof then output;
run;
*/
ods exclude none;
/* Perform steps below only if LOGISTIC MODEL on a bootstrap sample converges */
DATA _NULL_; Set ConvergenceStatus;
call symput('converged', status);
run;
%IF &converged = 0 %THEN %DO;
PROC APPEND BASE = BASE1 DATA = ScoreFitStat1;
run;
/*
PROC APPEND BASE = BASE3 DATA = NumberinModel;
run;
*/
PROC APPEND BASE = BASE4 DATA = Lift_Chart1;
run;
ods exclude all;
PROC LOGISTIC INMODEL = OM;
SCORE DATA = &DATASET FITSTAT
OUT=SCORED2(keep = &TARGET P_1);
ods output ScoreFitStat = ScoreFitStat2;
run;
ods exclude none;
PROC APPEND BASE = BASE2 DATA = ScoreFitStat2;
run;
```

```
* Create Lift Charts from Scoring Full Sample with Boot Model;
PROC RANK DATA= SCORED2 OUT= RANKOUT2
GROUPS=&GROUPS DESCENDING;
VAR P_1; /* variable that is ranked */
RANKS RANKP; /* name of ranks */
PROC MEANS DATA= RANKOUT2 NOPRINT;
CLASS RANKP; VAR P_1 &TARGET;
OUTPUT OUT= Lift_Chart2 MEAN= PREDICT &TARGET;
DATA Lift_Chart2; SET Lift_Chart2;
Replicate = &I;
run;
* END: Create Lift Charts from Scoring Full Sample with Bootstrap Model;
PROC APPEND BASE = BASE5 DATA = Lift_Chart2;
%END;
%END;
%MEND;

%REP(200);
```

The **%MACRO REP** creates all results. The code below combines these results and prints the report.

```
/* Merge Performance Stats from Bootstrap Model and scoring on full Model */
DATA BOTH; MERGE
BASE1(RENAME = (AUC=c_B BrierScore=ASE_B) DROP=Dataset)
BASE2(RENAME = (AUC=c_F BrierScore=ASE_F) DROP=Dataset)
;
d_B = 2*c_B - 1;
d_F = 2*c_F - 1;
Diff_d = d_B - d_F;
Diff_c = c_B - c_F;
Diff_ASE = ASE_B - ASE_F;
run;
PROC MEANS DATA = BOTH NOPRINT;
VAR Diff_d Diff_c Diff_ASE d_B d_F c_B c_F ASE_B ASE_F;
OUTPUT OUT = MEANOUT
MEAN = Diff_d Diff_c Diff_ASE d_B d_F c_B c_F ASE_B ASE_F;
run;
PROC PRINT DATA = MEANOUT;
VAR Diff_c Diff_ASE c_B c_F ASE_B ASE_F;
TITLE1 "MEANOUT ... Optimism Performance Statistics";
run;
/*
PROC FREQ DATA = BASE3; TABLES NumberinModel;
TITLE1 "Number of predictors in Bootstrap Models";
run;
*/
/* Compute optimism for Lift Charts */
DATA Base4_5; Merge Base4 Base5; by replicate _type_ rankP;
DROP _FREQ_ _TYPE_;
Optimism_Y = Y_B - Y;
Optimism_Predict = Predict_B - Predict;
* If RankP = . then RankP = -9;
run;
PROC MEANS DATA = Base4_5 NOPRINT MISSING;
Class RankP;
Var Optimism_Y Optimism_Predict Y_B Y Predict_B Predict;
OUTPUT OUT = Optimism_Lift(where=(_TYPE_ = 1))
N = N
MEAN = Optimism_Y Optimism_Predict Y_B Y Predict_B Predict;
PROC FORMAT;
value ALL . = "ALL";
PROC PRINT DATA = Optimism_Lift;
Var N _TYPE_ RankP Optimism_Y Optimism_Predict Y_B Y Predict_B Predict;
format RankP ALL.;
TITLE1 "Optimism for Lift Chart";
run;
TITLE; run;
```

SAS Code 14

Here is the report when the entire code is run with `%REP(2);` … only 2 bootstrap samples. (Try this first.)

MEANOUT … Optimism Performance Statistics

Obs	Diff_c	Diff_ASE	c_B	c_F	ASE_B	ASE_F
1	.004142554	-.001348403	0.627923	0.62378	0.237578	0.238927

c_F = Mean c-statistic computed on scored WORK by bootstrap models
c_B = Mean c-statistic computed on bootstrap samples by bootstrap models
ASE_F = Mean ASE computed on scored WORK by bootstrap models
ASE_B = Mean ASE computed on bootstrap samples by bootstrap models
Diff_c = c_B – c_F
Diff_ASE = ASE_B – ASE_F

Optimism Correction for Lift Chart with only 2 Bootstrap samples.

Obs	N	_TYPE_	RANKP	Optimism_Y	Optimism_Predict	Y_B	Y	Predict_B	Predict
1	2	1	ALL	-0.004000	-.004642660	0.49600	0.500	0.49600	0.50064
2	2	1	0	-0.020946	-.002454290	0.59505	0.616	0.63731	0.63976
3	2	1	1	0.029557	-.005139609	0.58956	0.560	0.52960	0.53474
4	2	1	2	-0.005945	-.002666565	0.47405	0.480	0.46303	0.46569
5	2	1	3	-0.013495	-.007552681	0.33051	0.344	0.35482	0.36237

Y = Mean of Y for WORK
Y_B = Mean of Y over bootstrap samples
Predict = Mean of P computed on scored WORK by bootstrap models
Predict_B = Mean of P computed on bootstrap samples by bootstrap models
Optimism_Y = Y_B – Y
Optimism_Predict = Predict_B – Predict

APPENDIX C: %MACRO AS_Simulation for Optimism Correction Study

SAS datasets output from %MACRO AS_Simulation are saved in a library with libref = DATA_SE

Here are parameters for: %MACRO AS_Simulation(Sample, Start_Seed, End_Seed, REPS);

Sample: This is "N" as in the description of the simulation study in this paper.

N is the size of the Analysis dataset 1000, 2000, 3000, 4000, 5000, 6000

Start_Seed and End_Seed: Used in: `%DO SEED = &START_SEED %TO &END_SEED;`
End-Seed - Start_Seed is the number of samples in this simulation run. The Start_Seed and End_Seed were introduced so that the targeted 50 simulations could be divided into smaller chunks to avoid excessive run-times

REPS is the replications of bootstrap samples in PROC SURVEYSELECT. REPS was 50 in the study.

F. Harrell suggests 200 but this led to excessive run times in my environment.

The interested user might try this: `%AS_Simulation(1000, 1, 3, 50)` to get the feel of the macro. This macro call creates three samples of N = 1000 + 22000 but runs 50 bootstrap samples for the optimism correction calculation. Even this macro call, with only 3 iterations, takes a moment or two to run.

Note: “AS” stands for Austin and Steyerberg who provided the motivation for this macro.

Here is the code. There is the main macro %AS_SIMULATION and a submacro %REP. Some features are disabled (see the commented out code).

```
options nomlogic nomprint;
%MACRO REP(R);
%IF %SYSFUNC(EXIST(BASE1)) = 1 %THEN %DO;
PROC DELETE DATA = BASE1;
run;
%END;
%IF %SYSFUNC(EXIST(BASE2)) = 1 %THEN %DO;
PROC DELETE DATA = BASE2;
run;
%END;
%IF %SYSFUNC(EXIST(BASE3)) = 1 %THEN %DO;
PROC DELETE DATA = BASE3;
run;
%END;
%IF %SYSFUNC(EXIST(ScoreFitStat1)) = 1 %THEN %DO;
PROC DELETE DATA = ScoreFitStat1;
run;
%END;
%IF %SYSFUNC(EXIST(ScoreFitStat2)) = 1 %THEN %DO;
PROC DELETE DATA = ScoreFitStat2;
run;
%END;
/* %IF %SYSFUNC(EXIST(NumberinModel)) = 1 %THEN %DO;
PROC DELETE DATA = NumberinModel;
run;
%END; */
%DO I = 1 %TO &R;
ods exclude all; /* Important -  this removes massive amount of printouts */
PROC LOGISTIC DATA = BootSamples(where=(replicate=&I))
desc OUTMODEL = OM;
FREQ NumberHits;
CLASS C12 C34;
MODEL Y (descending) = X1-X7 C12 C34;
SCORE DATA = BootSamples(where=(replicate=&I)) FITSTAT;
ods output ScoreFitStat = ScoreFitStat1;
ods output ConvergenceStatus = ConvergenceStatus;
* ods output ModelBuildingSummary=ModelBuildingSummary;
run;
/* DATA NumberinModel(keep=NumberinModel); Set ModelBuildingSummary end=eof;
if eof then output;
run; */
ods exclude none;
DATA _NULL_; Set ConvergenceStatus;
     call symput('converged',status);
run;
%IF &converged = 0 %THEN %DO;
     PROC APPEND BASE = BASE1 DATA = ScoreFitStat1;
     run;
     /*PROC APPEND BASE = BASE3 DATA = NumberinModel;
     run;    */
     ods exclude all;
     PROC LOGISTIC INMODEL = OM;
     SCORE DATA = SAMPLE FITSTAT;
     ods output ScoreFitStat = ScoreFitStat2;
     run;
     PROC APPEND BASE = BASE2 DATA = ScoreFitStat2;
     run;
     %END;
%END; /* %DO I = 1 %TO &R */
/* RUN this code to reduce WORK Library !! */
/* PROC DATASETS LIB=work NODETAILS;
DELETE DATA: ;
QUIT;
PROC DELETE DATA = OM;
run; */
ods exclude none;
%MEND REP;
```

```
%MACRO AS_Simulation(Sample, Start_Seed, End_Seed, REPS);

%IF %SYSFUNC(EXIST(XFREQ_Y1)) = 1 %THEN %DO;
PROC DELETE DATA = XFREQ_Y1;
run;
%END;
%IF %SYSFUNC(EXIST(XEXTERNAL)) = 1 %THEN %DO;
PROC DELETE DATA = XEXTERNAL;
run;
%END;
%IF %SYSFUNC(EXIST(XAPPARENT)) = 1 %THEN %DO;
PROC DELETE DATA = XAPPARENT;
run;
%END;
%IF %SYSFUNC(EXIST(XSPLIT)) = 1 %THEN %DO;
PROC DELETE DATA = XSPLIT;
run;
%END;
%IF %SYSFUNC(EXIST(XOPTIMISM)) = 1 %THEN %DO;
PROC DELETE DATA = XOPTIMISM;
run;
%END;

%DO SEED = &START_SEED %TO &END_SEED;

   DATA SAMPLE POPULATION;
   do ID = 1 to &Sample + 22000;
   PART = MOD(ID,2); /* use later */
   ALL = 0;
   EXTERNAL = (ID > &Sample);
   cumLogit = ranuni(&Seed);
   e = 1*log( cumLogit/(1-cumLogit ));
   X1 = rannor(&Seed);
   X2 = rannor(&Seed);
   X3 = rannor(&Seed);
   X4 = rannor(&Seed);
   X5 = rannor(&Seed);
   X6 = rannor(&Seed);
   X7 = X6*X5;
   B1 = (ranuni(&Seed) < .4);
   B2 = (ranuni(&Seed) < .6);
   B3 = (ranuni(&Seed) < .5);
   B4 = (ranuni(&Seed) < .5);
   C12 = B1 + B2;
   C34 = B3 + B4;
   Z = -6 + X1**2 + log(X2+8) + .01*X3 + 2*X7 + 0.1*B1 + B2 + B3 + B4 + 2*e;
   Y = (Z > 5.0);
   IF ID <= &SAMPLE THEN OUTPUT SAMPLE;
   OUTPUT POPULATION;
   end;
   run;
   ODS exclude ALL;
   ODS OUTPUT OneWayFreqs = OneWayFreqs;
   PROC FREQ DATA = POPULATION(OBS=&SAMPLE); TABLES Y;
   run;
   DATA FREQ_Y1; SET OneWayFreqs;
   KEEP Frequency Percent Seed Sample_n EPP;
     if _N_ = 2;
     Seed = &Seed;
     Sample_n = &Sample;
     EPP = round(Frequency / 11,1);
   run;
   PROC APPEND BASE= XFREQ_Y1 NEW= FREQ_Y1;
   run;
   /* PROC PRINT DATA = FREQ_Y1;
   VAR Seed Sample_n Frequency Percent EPP;
   run; */

   ODS OUTPUT PartFitStats = PartFitStats(rename=(Validation = E_c_Stat));
   PROC HPLOGISTIC DATA= POPULATION;
   PARTITION ROLE=EXTERNAL (TRAIN="0", VALIDATE="1");
```

```
CLASS C12 C34;
MODEL Y (descending) = X1-X7 C12 C34;
SELECTION METHOD=NONE;
run;
DATA EXTERNAL; SET PartFitStats(OBS=1);
KEEP Source E_c_Stat Seed Sample_n;
  Seed = &Seed;
  Sample_n = &Sample;
  Source = "EXTERNAL";
run;
PROC APPEND BASE= XEXTERNAL NEW= EXTERNAL;
/* PROC PRINT DATA = EXTERNAL;
VAR Source Seed Sample_n E_c_Stat;
run; */
run;
ODS OUTPUT PartFitStats = PartFitStats(rename=(TRAINING = A_c_Stat));
PROC HPLOGISTIC DATA= SAMPLE;
PARTITION ROLE=ALL (TRAIN="0", VALIDATE="1");
CLASS C12 C34;
MODEL Y (descending) = X1-X7 C12 C34;
SELECTION METHOD=NONE;
run;
DATA APPARENT; SET PartFitStats(OBS=1);
KEEP Source A_c_Stat Seed Sample_n;
  Seed = &Seed;
  Sample_n = &Sample;
  Source = "APPARENT";
run;
PROC APPEND BASE= XAPPARENT NEW= APPARENT;
/* PROC PRINT DATA = APPARENT;
VAR Source Seed Sample_n A_c_Stat;
run; */
run;

ODS OUTPUT PartFitStats = PartFitStats(rename=(VALIDATION = S_c_Stat));
PROC HPLOGISTIC DATA= SAMPLE;
PARTITION ROLE=PART (TRAIN="0", VALIDATE="1");
CLASS C12 C34;
MODEL Y (descending) = X1-X7 C12 C34;
SELECTION METHOD=NONE;
run;
DATA SPLIT; SET PartFitStats(OBS=1);
KEEP Source S_c_Stat Seed Sample_n;
  Seed = &Seed;
  Sample_n = &Sample;
  Source = "SPLIT   ";
run;
PROC APPEND BASE= XSPLIT NEW= SPLIT;
/* PROC PRINT DATA = SPLIT;
VAR Source Seed Sample_n S_c_Stat;
run; */
run;
ODS exclude NONE;

PROC SURVEYSELECT DATA=SAMPLE
OUT=BootSamples
NOPRINT SEED=111
METHOD=urs /* with replacement */
SAMPRATE=1
REPS=200;
run;

%REP(&REPS);

DATA BOTH; MERGE
  BASE1(RENAME = (AUC=c_B) DROP=Dataset)
  BASE2(RENAME = (AUC=c_F) DROP=Dataset)
  ;
  Diff_c = c_B - c_F;
run;
PROC MEANS DATA = BOTH NOPRINT;
```

```
    VAR Diff_c c_B c_F;
    OUTPUT OUT = MEANOUT
    MEAN = Diff_c c_B c_F;
    run;
    DATA OPTIMISM; SET MEANOUT;
      Sample_n = &SAMPLE;
      Seed = &SEED;
    PROC APPEND BASE = XOPTIMISM NEW = OPTIMISM;
    TITLE2;
    run;

    %END; /* %DO SEED = &START_SEED %TO &END_Seed; */

DATA DATA_SE.Report3_&Sample._&START_SEED._&END_SEED;
MERGE XFREQ_Y1 XEXTERNAL XAPPARENT XSPLIT XOPTIMISM; by Sample_n Seed;
DROP SOURCE;
O_c_Stat = A_c_Stat - Diff_c;
run;
PROC PRINT DATA = DATA_SE.Report3_&Sample._&START_SEED._&END_SEED;
VAR Sample_n Seed EPP E_c_Stat A_c_Stat S_c_Stat O_c_Stat;
run;

PROC MEANS DATA = DATA_SE.Report3_&Sample._&START_SEED._&END_SEED NOPRINT;
CLASS EPP;
VAR E_c_Stat A_c_Stat S_c_Stat O_c_Stat Diff_c;
OUTPUT OUT = DATA_SE.Report3_&Sample._&START_SEED._&END_SEED._SUM
MEAN = E_c_Stat A_c_Stat S_c_Stat O_c_Stat Diff_c;
run;
PROC PRINT DATA = DATA_SE.Report3_&Sample._&START_SEED._&END_SEED._SUM;
VAR _FREQ_ EPP E_c_Stat A_c_Stat S_c_Stat O_c_Stat Diff_c;
TITLE1 "AS_Simulation(&Sample, &Start_Seed, &End_Seed, &Reps)";
TITLE2 "Diff_c = bootstrap_c - full_c";
run;

%MEND;
```

SAS Code 15

To include more iterations (the final target is 50), the code below can be used to combine the results of small runs of **%AS_Simulation()**. This “chunking” may be necessary to avoid excessive run times.

Here, N = 1000. After Setting together the two datasets in the SET statement, the start and end seeds go from 1 to 4

Note that a libname reference “DATA_SE” was assigned.

```
DATA DATA_SE.report3_1000_1_4;
SET DATA_SE.report3_1000_1_3 DATA_SE.report3_1000_4_4;
run;
PROC MEANS DATA = DATA_SE.report3_1000_1_4 NOPRINT;
CLASS EPP;
VAR E_c_Stat A_c_Stat S_c_Stat O_c_Stat Diff_c;
OUTPUT OUT = DATA_SE.report3_1000_1_4_SUM
MEAN = E_c_Stat A_c_Stat S_c_Stat O_c_Stat Diff_c;
run;
PROC PRINT DATA = DATA_SE.report3_1000_1_4_SUM;
VAR _FREQ_ EPP E_c_Stat A_c_Stat S_c_Stat O_c_Stat Diff_c;
TITLE1 "AS_Simulation(1000, 1, 4, 50)";
TITLE2 "Diff_c = bootstrap_c - full_c";
run;
```

SAS Code 16

See the report below:

AS_Simulation(1000, 1, 4, 50)
Diff_c = bootstrap_c - full_c

Obs	_FREQ_	EPP	E_c_Stat	A_c_Stat	S_c_Stat	O_c_Stat	Diff_c
1	4		0.7129	0.7645	0.7068	0.72405	0.040468
2	1	5	0.6934	0.7684	0.7384	0.72212	0.046329
3	2	7	0.7147	0.7727	0.6885	0.73921	0.033443
4	1	8	0.7288	0.7443	0.7116	0.69563	0.048657

Figure 15.

APPENDIX D: REGRESSION SPLINES AND OPTIMISM CORRECTION

Here, again, is dataset WORK:

```
%LET SEED = 3; %LET N = 1000;
DATA WORK;
call streaminit(&SEED);
DO ID = 1 to &N;
     if mod(ID,2) = 0 then Y = 1;
     else Y = 0;
     if Y = 1 then do;
            X1 = rand('normal') + .2;
            X2 = rand('normal') + .2;
            end;
     if Y = 0 then do;
            X1 = rand('normal') - .2;
            X2 = rand('normal') - .2;
            end;
     output;
     end;
run;
```

SAS Code 17

Now “Natural Cubic Splines” are added to WORK to form dataset WORK2. The spline columns are saved in dataset Spline_Design and then merged to WORK to form dataset WORK2. X1_spl1 is the same as X1 and, likewise, for X2_spl1. These redundant variables are dropped in the MERGE.

```
ODS EXCLUDE ALL;
PROC LOGISTIC DATA = WORK desc
OUTDESIGN = Spline_Design; /* spline design matrix */
EFFECT X1_spl = spline( X1 / details naturalcubic basis=tpf(noint)
knotmethod=PERCENTILES(4));
EFFECT X2_spl = spline( X2 / details naturalcubic basis=tpf(noint)
knotmethod=PERCENTILES(4));
MODEL Y = X1_spl X2_spl;
run;
ODS EXCLUDE NONE;
DATA WORK2; MERGE WORK Spline_Design;  /* No BY statement needed */
drop X1_spl1 X2_spl1 INTERCEPT;
PROC PRINT DATA = Spline_Design (obs=3);
PROC PRINT DATA = WORK2 (obs=3);
/* Instead of PROC LOGISTIC, PROC HPLOGISTIC or PROC HPGENSELECT could be used. */
PROC LOGISTIC DATA = WORK2 desc;
MODEL Y = X1 X2 X1_spl: X2_spl:
/ SELECTION=BACKWARD SLS=.05;
run;
```

SAS Code 18

The second PROC LOGISTIC creates the Apparent Model.

Similar PROC LOGISTIC's would appear in the Optimism Correction coding, with distinct logistic models being created for each bootstrap sample.

Here is the output from SAS Code 18.

Obs	Y	Intercept	X1_spl1	X1_spl2	X1_spl3	X2_spl1	X2_spl2	X2_spl3
1	0	1	1.53921	4.30840	1.96700	1.22129	3.83661	1.50260
2	1	1	0.91711	2.47053	1.02457	0.27773	0.90760	0.16916
3	0	1	-0.97577	0.00000	0.00000	0.21524	0.76839	0.11995

Obs	ID	Y	X1	X2	X1_spl2	X1_spl3	X2_spl2	X2_spl3
1	1	0	1.53921	1.22129	4.30840	1.96700	3.83661	1.50260
2	2	1	0.91711	0.27773	2.47053	1.02457	0.90760	0.16916
3	3	0	-0.97577	0.21524	0.00000	0.00000	0.76839	0.11995

Much of the output was omitted from the PROC LOGISTIC report. Here is the final portion:

Analysis of Maximum Likelihood Estimates

Parameter	DF	Estimate	Standard Error	Wald Chi-Square	Pr > ChiSq
Intercept	1	0.1795	0.0936	3.6802	0.0551
X1	1	0.5864	0.1257	21.7748	<.0001
X2	1	0.2975	0.0661	20.2361	<.0001
X1_spl3	1	-0.4020	0.1633	6.0587	0.0138

Odds Ratio Estimates

Effect	Point Estimate	95% Wald Confidence Limits	
X1	1.798	1.405	2.300
X2	1.347	1.183	1.533
X1_spl3	0.669	0.486	0.921

Association of Predicted Probabilities and Observed Responses

Percent Concordant	62.7	Somers' D	0.254
Percent Discordant	37.3	Gamma	0.254
Percent Tied	0.0	Tau-a	0.127
Pairs	250000	c	0.627

Figure 16.

Notice that X1_spl3 was added to the model. Of course, this was simply luck for this example.