Paper 4647-2020 (Submitted to SAS Global Forum)

Simple and Efficient Bootstrap Validation of Predictive Models Using SAS/STAT® Software

Isaiah Lankham, University of California Office of the President, Oakland, CA

Matthew Slaughter, Kaiser Permanente Center for Health Research, Portland, OR

ABSTRACT

Validation is essential for assessing a predictive model's performance with respect to optimism or overfitting. While traditional sample-splitting techniques like cross validation require us to divide our data between model building and model assessment, bootstrap validation enables us to use the full sample for both. This paper demonstrates a simple method for efficiently calculating bootstrap-corrected measures of predictive model performance using SAS/STAT® procedures. While several SAS® procedures have options for automatic cross validation, bootstrap validation requires a more manual process. Examples focus on logistic regression using the LOGISTIC procedure, but these techniques can be readily extended to other procedures and statistical models.

The running example can be downloaded in Jupyter Notebook and .sas file format from https://github.com/saspy-bffs/sgf-2020-bootstrap-validation

INTRODUCTION

The term *overfitting* is broadly used when a statistical model intended to make general predictions about a population instead narrowly describes the unique variations present only in the sample dataset used to train it (see, e.g., [9], [11], and [30]). One common cause for overfitting is using a sample dataset that's insufficiently representative of the population it was drawn from. For the purposes of this paper, though, we only consider the other common cause for overfitting, which is building a model with spurious predictors or too many predictors.

Specifically, we study a measure for overfitting called *optimism*, which we'll assess using techniques based on the bootstrap [5]. Optimism is calculated by taking the difference between the apparent performance of a model and its performance when predictions are made on new data [9]. However, we don't actually need to have new data. Instead, we can estimate the optimism by repeating our modeling process (including any automated variable selection) on bootstrap samples drawn from our sample dataset.

Broadly speaking, there are two broad categories of validation techniques for predictive models, internal and external [9]. *Internal validation* consists of various techniques for estimating the generalizability of a model using only the data available during model development [30]. This typically involves partitioning or resampling from the available data to simulate potential variability in the wider population. The most common techniques include split-sample validation and various types of cross-validation [13], as well as our topic of interest, the bootstrap. (See Sections 1 and 3 for complete definitions.)

If data are also available from a different population, setting, or time period, we can also perform *external validation* [30]. External validation is important because it allows us to apply scientific standards of reproducibility and to measure the performance of the model in actual practice. However, it's not a substitute for internal validation. Even when external data are available, internally validated measures of model performance are important. A model that hasn't been validated internally is unlikely to improve when validated externally.

SECTION 1. THE BOOTSTRAP VALIDATION ALGORITHM

First proposed by Bradley Efron [5], the bootstrap is a non-parametric method for estimating the sampling distribution of a statistic by resampling with replacement from available data. Efron, Gong, and Tibshirani (see [2], [3], [4], [7], and [8]) later explored how the original bootstrap could be adapted to calculate nearly unbiased and relatively stable estimates of optimism for a specific model performance metric. By subtracting the optimism from the performance metric, we can obtain a "corrected" version, which better accounts for possible overfitting. As has been shown through numerous simulation studies, this method tends to produce much more accurate estimates of true model performance than simply obtaining bootstrap estimates of performance metrics directly (see, e.g., [9] and [30]).

The specific steps for Bootstrap Validation are as follows, which we illustrate with a detailed example in Section 2 below:

1. Train a model on a sample dataset, and record the value of a performance metric of interest.
2. Form sufficiently many bootstrap samples[1] by drawing randomly with replacement from the original sample dataset.
3. Train a new model[2] on each bootstrap sample, and record each corresponding value of the performance metric for each bootstrap-sample-derived model.
4. Apply each bootstrap-sample-derived model to the original sample dataset, and measure the performance metric.
5. Estimate optimism by taking the mean of the differences between the values calculated in Step 3 (the apparent performance of each bootstrap-sample-derived model) and Step 4 (each bootstrap-sample-derived model's performance when tested on the original sample).
6. Calculate the optimism-corrected value of the performance metric as the difference between the values calculated in Step 1 (the naïve value) and Step 5 (the estimated optimism).

[1] Steyerberg reports that 100-200 bootstrap replicates are often sufficient [30]. However, one simulation study found that stability continued to improve in small datasets up to 500 bootstraps [29]. More bootstraps are recommended for smaller sample sizes, but the choice may be limited by the available computing resources.

[2] All steps of the original modeling process, including any automated variable selection, should be repeated when models are trained on each bootstrap sample.

Obs	RIDAGEYR	DMQ051	DMD110	INDHHINC	INDFMPIR	BPQ100D	LBXTC	BMXBMI	LBXTR	hyper
1	61	2	3	12	3.33	2	285	23.20	210	0
2	85	2	1	6	2.71	2	276	30.67	181	1
3	30	2	1	11	4.19	2	254	25.60	308	0
4	59	2	1	6	2.58	1	174	26.17	147	0
5	41	2	1	7	2.59	1	183	41.93	341	0
6	76	1	1	6	2.40	1	184	31.30	123	1
7	66	2	1	8	4.52	1	226	26.48	114	0
8	41	2	1	8	2.17	1	258	26.40	272	0
9	67	2	1	3	0.73	1	232	28.78	541	1
10	59	2	1	3	1.21	1	202	26.10	239	0

Figure 1. The first few rows of the sample dataset `example_dataset` used in the running example in Section 2.

SECTION 2. A SIMPLE AND EFFICIENT SAS IMPLEMENTATION

While many statistical procedures in SAS have built-in options for data partitioning (e.g., the PARTITION statement in PROC HPLOGISTIC [23]) or cross-validation (e.g., the CVMETHOD= options in PROC GLMSELECT [22]), none appear to be available for bootstrap estimation of optimism as of SAS version 9.4M6[3]. The example below illustrates how SAS language tools for iteration across groups in datasets can be used instead.

For this example, we use a combination of the LOGISTIC [24] and SURVEYSELECT [26] procedures from SAS/STAT, as well as the SQL procedure [19] and the DATA step [18] from Base SAS. We also use ODS OUTPUT [15] statements to capture output from PROC LOGISITIC, as well as additional ODS statements to suppress output during bootstrapping as recommended by Rick Wicklin [33]. In addition, we use macro variables to store the names of repeated variable lists.

Specifically, we use PROC LOGISTIC to predict a binary outcome (hypertension[4]) from various vital measures and demographic characteristics, and we use *concordance*[5] (aka *C-statistic* or *AUC*) as our performance metric of interest. Our sample dataset is based on publicly available data from a CDC tutorial [14], with the first few rows shown in Figure 1. All steps used to create the example dataset, as well as the full example itself, can be found at https://github.com/saspy-bffs/sgf-2020-bootstrap-validation.

[3] For a full list of SAS procedures with options for resampling see http://support.sas.com/kb/22/220.html.

[4] *Hypertension* is a medical term for a condition in which a patient suffers from unusually high blood pressure, often diagnosed when multiple high blood pressure readings are recorded over several weeks [31]. For this example, we define hypertension based on an average systolic blood pressure over 140, an average diastolic reading over 90, or a patient taking prescribed antihypertensive medication. We also use as predictors patient BMI, age, cholesterol/triglyceride levels, country of birth, service in the armed forces, annual household income, and substance abuse. (Note: This model is presented purely to illustrate SAS programming techniques. It is not intended as a serious or effective diagnostic tool.)

[5] For binary outcomes, the C-statistic is equivalent to the area under the receiver operating curve and represents the probability that a patient with an outcome is given a higher probability by the model than a random patient without the outcome. See [30] for a full overview.

Association of Predicted Probabilities and Observed Responses			
Percent Concordant	70.0	Somers' D	0.400
Percent Discordant	30.0	Gamma	0.400
Percent Tied	0.0	Tau-a	0.178
Pairs	7021	c	0.700

Figure 2. Part of the output from PROC LOGISTIC when training the initial model in Step 1 of the running example.

RUNNING EXAMPLE — STEP 1: TRAIN A MODEL

We begin by training our initial model using a typical PROC LOGISTIC step preceded by a common ODS trick:

```
* SAS Code producing the output shown in Figure 2;
ods output Association=model_association_table( ❶
    where=(Label2='c') ❷
    keep=Label2 nValue2 ❷
    rename=(nValue2=original_model_c_statistic) ❷
);
proc logistic data=example_dataset; ❸
    class &class_variables.; ❹
    model &outcome. = &predictor_variables.; ❹
run;
```

The ODS OUTPUT statement in ❶ captures the table of association measures produced by ❸–❹ and shown in Figure 2, and the naïve C-statistic is saved in a dataset called model_association_table using ODS OUTPUT with dataset options in ❷. (For an explanation of how lines ❶–❷ were constructed, see the Appendix at the end of this paper.) Also, to keep the example code concise, the following macro variables have been used to encapsulate variables from the original source CDC dataset:

```
%let outcome = hyper(EVENT='1');
%let class_variables = bpq100d dmq051 dmd110;
%let predictor_variables = lbxtc bpq100d bmxbmi ridageyr lbxtr dmq051
dmd110 indhhinc indfmpir;
```

Input Data Set	EXAMPLE_DATASET
Random Number Seed	1354687
Sampling Rate	1
Sample Size	178
Expected Number of Hits	1
Sampling Weight	1
Number of Replicates	500
Total Sample Size	89000
Output Data Set	BOOTSTRAP_SAMPLES

Figure 3. The output of PROC SURVEYSELECT when used to construct bootstrap samples in Step 2 of the running example.

RUNNING EXAMPLE — STEP 2: GENERATE BOOTSTRAP SAMPLES

Bootstrap samples, which should be drawn randomly with replacement from the original sample dataset, can be generated using PROC SURVEYSELECT as follows:

```
* SAS Code producing the output shown in Figure 3;
proc surveyselect
        data=example_dataset
        out=bootstrap_samples ❶
        seed=1354687 ❷
        method=urs ❸
        outhits ❹
        rep=500 ❺
        samprate=1 ❻
    ;
run;
```

The dataset bootstrap_samples created in ❶ containing the bootstrap samples, which were generated using random sampling based on the seed[6] specified in ❷. In addition, the options in ❸ and ❹ ensure elements are drawn with equal probability and with replacement. (In other words, selecting the same observation more than once will result in distinct observations in a bootstrap sample.) Finally, the options in ❺ and ❻ specify that 500 bootstrap samples (following the findings in [29]) of the same size as the original dataset should be formed.

Note that while it's also possible to generate random samples with a DATA step [34], PROC SURVEYSELECT is custom-made for straightforward sample-selection. However, a DATA step could potentially be more efficient, especially when sample sizes are very large [28].

[6] The seed is totally arbitrary and only serves to make the results reproducible.

Obs	Replicate	Label2	c_statistic_value
1	1	c	0.732240
2	2	c	0.701415
3	3	c	0.690810
4	4	c	0.731947

Figure 4. The first few rows of the dataset `bootstrap_association_table` created in Step 3 of the running example, when C-statistics are calculated for the models trained on each bootstrap sample in the running example.

RUNNING EXAMPLE — STEP 3: TRAIN MODELS IN EACH BOOTSTRAP

Now that we have our bootstrap samples, it's time to train models using PROC LOGISITIC with a BY statement. We also suppress output as the 500 logistic regression models are created:

```
* turn off all output;
ods graphics off; ❶
ods exclude all; ❶
ods noresults; ❶

* SAS Code producing the dataset shown in Figure 4;
ods output Association=bootstrap_association_table( ❷
    where=(Label2='c') ❸
    keep=Replicate Label2 nValue2 ❸
    rename=(nValue2=c_statistic_value) ❸
);
proc logistic data=bootstrap_samples outmodel=bootstap_models; ❹
    by Replicate; ❺
    class &class_variables.; ❻
    model &outcome. = &predictor_variables.; ❻
run;
```

After opening a "no output sandwich" in ❶, which we close in the Step 4 below, C-statistics for each of the 500 models created in ❹–❻ are captured in output dataset `bootstrap_association_table` in ❷ using the dataset options in ❸. (As a reminder, the technique for constructing ❷–❸ can be found in the Appendix at the end of this paper.) We also use the OUTMODEL option in ❹ to capture the models created[7] by iterating over the bootstrap samples with the BY statement in ❺.

It's important to note that the PROC LOGISTIC step in ❹–❻ is identically to the one used to train our original mode in Step 1 above. In addition, note how easily the `Replicate` column created by PROC SURVEYSELECT allows us to iterate over the bootstrap samples without a macro loop. Due to the increased overhead of starting and stopping PROC LOGISITIC repeatedly, a macro loop would most likely be significantly less efficient [32].

[7] We could also use a CODE or STORE statement, but it would change how the saved models are processed later.

Obs	Replicate	c_statistic_value
1	1	0.616009
2	2	0.666714
3	3	0.652898
4	4	0.683378

Figure 5. The first few rows of the dataset `bootstrap_scores` created in Step 4 of the running example, when the models trained on bootstrap samples in Step 3 are evaluated using the original sample dataset. (Note that the C-statistics tend to be lower for each `Replicate` when compared to Figure 4, suggesting possible overfitting.)

RUNNING EXAMPLE — STEP 4: TEST BOOTSTRAP MODELS

We now test each of the 500 models created in Step 3 with one final PROC LOGISTIC step using the previously saved dataset of logistic models:

```
* SAS Code producing the dataset shown in Figure 5;
ods output Scorefitstat=bootstrap_scores( ❶
    keep=Replicate AUC ❷
    rename=(AUC=c_statistic_value) ❷
);
proc logistic inmodel=bootstap_models; ❸
    score ❹
        data=example_dataset ❹
        fitstat ❹
    ;
    by Replicate; ❺
run;

* turn output back on;
ods results; ❻
ods select all; ❻
ods graphics on; ❻
```

C-statistics[8] for each of the 500 models created in Step 3 are captured in output dataset bootstrap_scores in ❶ using the dataset options in ❷. (As a reminder, the technique for constructing ❶–❷ can be found in the Appendix at the end of this paper.) Rather than specifying an input dataset in ❸, we instead load the models previously we trained in Step 3, and we use a SCORE statement in ❹ to apply each of the 500 models to the original dataset sample.

We also close the "no output sandwich" from Step 3 in ❻ since we've finished iterating over our bootstrap samples.

[8] C-statistics are referred to as Area Under the Curve (or AUC) in this example for technical reasons beyond the scope of this paper. See [30] for an overview.

Obs	optimism
1	0.072888

Figure 6. The dataset `model_optimism` created in Step 5 of the running example.

RUNNING EXAMPLE — STEP 5: ESTIMATE OPTIMISM

We're now ready to calculate the optimism of each model trained on a bootstrap sample, with the mean of these values being an estimate for the optimism of the original model created in Step 1:

```
* SAS Code producing the dataset shown in Figure 6;
proc sql;
    create table model_optimism as ❶
        select
            avg(A.c_statistic_value-B.c_statistic_value) as optimism ❷
        from
            bootstrap_association_table as A ❸
            inner join ❹
            bootstrap_scores as B ❺
            on A. Replicate = B. Replicate ❻
    ;
quit;
```

Here, a PROC SQL step is used to simultaneously accomplish the following tasks:

- Build a new dataset in ❶, which stores the estimated optimism for the model from Step 1.
- Compute the optimism for each of the models built from bootstrap samples in Step 3 by taking its naïve C-statistic and subtracting the scored C-statistic calculated in Step 4 (when the models were applied to the original sample dataset). In particular, the datasets formed in Steps 3 and 4 are joined on `Replicate` in ❸–❻, allowing us to calculate the mean of their row-by-row differences in ❷.

Note that while we could also have accomplished these same tasks using a combination of DATA steps and other PROC steps, PROC SQL makes it straightforward to quickly estimate optimism in a single program step.

Naive C-Statistic	Optimism	Optimism-Corrected C-Statistic
0.700185	0.072888	0.62730

Figure 7. The dataset corrected_model_evaluation created in Step 6 of the running example, when the naïve C-statistic computed in Step 1 is corrected using the optimism computed in Step 5.

RUNNING EXAMPLE — STEP 6: ADJUST PERFORMANCE W/ OPTIMISM

And finally, we're ready to use the estimated optimism from Step 5 to adjust the naïve C-statistic calculated in Step 1:

```
* SAS Code producing the output shown in Figure 7;
data corrected_model_evaluation; ❶
    set model_association_table; ❷
    set model_optimism; ❸

    corrected_c_statistic = original_model_c_statistic - optimism; ❹

    label ❺
        original_model_c_statistic = 'Naive C-Statistic' ❺
        optimism = 'Optimism' ❺
        corrected_c_statistic = 'Optimism-Corrected C-Statistic' ❺
    ;

    keep original_model_c_statistic optimism corrected_c_statistic; ❻
run;
proc print ❼
        data=corrected_model_evaluation ❼
        noobs ❼
        label ❼
    ;
run;
```

Here, we create a new dataset in ❶ by combining datasets from Steps 1 and 5 in ❷–❸. We then compute the optimism-corrected C-statistic for the model built in Step 1 by subtracting the estimated optimism from Step 5. Finally, we make output easier to read with a LABEL statement in ❺ and a KEEP statement in ❻ before printing the resulting dataset in ❼.

EXTENSIONS TO OTHER PERFORMANCE MEASURES OR MODELS

The method demonstrated in the running example is readily generalizable. Per the Appendix at the end of this paper, ODS TRACE can be used to determine the appropriate ODS OUTPUT statements for capturing many performance measures other than the C-statistic.

In addition, the running example can be extended to other models and SAS procedures, despite their varying syntax. When using a PROC that doesn't support the OUTMODEL and INMODEL options (e.g., PROC GLM), a CODE statement and subsequent DATA step can be used to make predictions on test data (see, e.g., [20] and [21]). Alternatively, the STORE statement could be used with a subsequent invocation of PROC PLM [25].

SECTION 3. RELATIVE ADVANTAGES OF THE BOOTSTRAP OVER OTHER VALIDATION TECHNIQUES

Compared to alternate internal validation techniques like split-sample and cross validation, which are already implemented in SAS, the bootstrap has certain distinct advantages.

Split-sample validation involves partitioning the available data into two or three sub-samples, training the model on one sample and evaluating its performance using the others (see, e.g., [9] , [13], and [30]). The main advantage is less intensive computation as the model only needs to be trained once. Because large numbers of observations are not used in the modeling process, though, the model that gets validated may be substantially different than the model that would have been produced using all of the available data.

When deciding about whether to deploy a model, this may mean we lack important information. Additionally, if the validation sample is chosen randomly, the estimates of performance may have high variance, even if they are unbiased on average. In other words, repeating the modeling process might yield substantially different results! However, these disadvantages are reduced as sample sizes grow, which is why split-sample validation is only recommended for extremely large sample datasets. Also, split-sample validation does not rely on subsamples being chosen randomly. For example, data from an earlier time period can be used for development, and data from a later time period can be reserved for validation. This allows split-sample validation to potentially measure the effect of non-random variation in the source data, which bootstrap validation does not.

Cross validation is an extension of split-sample techniques [11]. Multiple versions of the model are trained while leaving out a different subset each time, and model performance is measured on each omitted subset. Taking the average of the performance in each of these *folds* yields a cross-validated estimate of model performance. The number of folds can be increased until only a single observation is left out, which is equivalent to a procedure known as the *jack-knife*. Cross-validation allows a larger portion of the available data to be used in training the model than simple split-sample, and simulation studies have shown that cross-validation often needs to be replicated multiple times with different random splits to produce truly stable validation results [30]. Merely increasing the number of folds does reduce bias in the cross-validation estimator, but also decreases stability as training subsets become more similar. When the number of folds is large, cross-validation may underestimate variation in the underlying population, leading to misleadingly consistent variable selection in the model. Meanwhile, cross-validation with a small number of folds still holds back a significant amount of data from the modeling process.

Relative to these two two methods, the bootstrap is unique in allowing the entire sample to be used for both model development and model validation, while still providing nearly unbiased estimates of model performance [9]. Also, unlike split-sample and cross validation, bootstrapping allows us to estimate optimism and also gauge overfitting. Optimism-corrected estimates of performance are relatively stable compared to estimates produced by other resampling techniques because the bootstrap samples vary widely in composition and use the full sample size. This variability also helps the bootstrap appropriately model variation in variable selection.

There is evidence that bootstrap estimates can be biased when the size of the training data is small relative to the number of predictors (see [9] and [20]). However, even in cases where cross-validation estimates are less biased, Efron demonstrated that they also have much higher variance than bootstrap estimates of optimism-corrected performance [6]. Efron and Tibshirani have also described two bootstrap variants (known as the .632 and .632+ methods; see [4] and [6]) that may be less biased in such situations. These modifications are especially useful in very high-dimensional settings, such as genetic data (see [12] and [27]). Simulations have also shown that Efron's original bootstrap produces less biased estimates than either variant with 30 predictors and 200 observations (see [9] and [10]). In fact, Breiman [1] has shown that the bootstrap is as effective as having a separate test sample twice the size of the training data.

The main downside of the bootstrap is that it can be relatively resource-intensive when validating complex machine learning models on large datasets since all modeling steps must be repeated many times. Cross-validation, in particular, will often be less computationally intensive than the bootstrap when the number of folds is not large [30].

APPENDIX: ODS TRACE AND ODS OUTPUT

The examples in this paper make heavy use of the ODS OUTPUT statement [16] for capturing output objects. The basic syntax is as follows:

```
ods output <output object name> = <dataset name>;
```

Any output object with matching name will then be captured in the specified dataset. However, the names of the output objects produced by a procedure are usually not obvious. To discover them, it's helpful to use the ODS TRACE statement [17].

Repeating the PROC LOGISTIC step from Section 2, we can create an "ODS TRACE sandwich" as follows, toggling object-name reporting in the log:

```
ods trace on;
proc logistic data=example_dataset;
    class &class_variables.;
    model &outcome. = &predictor_variables.;
run;
ods trace off;
```

This will create a long string of log entries, including the following:

```
Output Added:
-------------
Name:        Association
Label:       Association Statistics
Template:    Stat.Logistic.Association
Path:        Logistic.Association
```

We can then capture the contents of the named output `Association` in a dataset using ODS OUTPUT before the same PROC LOGISITC step:

```
* SAS Code producing the dataset shown in Figure 8;
ods output Association=model_association_table;
proc logistic data=example_dataset;
    class &class_variables.;
    model &outcome. = &predictor_variables.;
run;
```

Obs	Label1	cValue1	nValue1	Label2	cValue2	nValue2
1	Percent Concordant	70.0	70.018516	Somers' D	0.400	0.400370
2	Percent Discordant	30.0	29.981484	Gamma	0.400	0.400370
3	Percent Tied	0.0	0	Tau-a	0.178	0.178442
4	Pairs	7021	7021.000000	c	0.700	0.700185

Figure 8. The dataset `model_association_table` created using an ODS OUTPUT statement to capture the named output `Association` generated by PROC LOGISITIC. Even though the column names aren't terribly descriptive, we can see that the fourth row contains information about the C-statistic, and that column `nValue2` gives the numerical value we're interested in capturing.

Note that objects should be listed in the log in the order they're displayed in other ODS destinations, and that they usually have names similar to the headings of the tables they capture. In Section 2, we captured the table labeled *Association of Predicted Probabilities and Observed Responses* (see Figure 2), which corresponds to output object `Association`. Once this information has been discovered, we are ready to capture the output in a dataset.

CONCLUSION

Internal validation is an essential best practice for statisticians, programmers, and researchers developing predictive models. Various sample-splitting or resampling techniques can be used, but the bootstrap in particular is appealing for producing stable and nearly unbiased estimates of model performance using all available data. No validation technique is ideal for all scenarios, though, so the analyst must make decisions based on the characteristics of the available data, the modeling techniques to be used, and the available computing resources. However, bootstrapping has been shown to generally be an effective and precise validation technique. While no SAS options currently exist to perform this procedure, existing SAS tools for resampling and iteration make it relatively straightforward to implement without resorting to complex macro programming.

REFERENCES

[1] Breiman, Leo. (1992) "The Little Bootstrap and Other Methods for Dimensionality Selection in Regression: X-Fixed Prediction Error." *Journal of the American Statistical Association*, 87(419), 738-754. Available at https://doi.org/10.2307/2290212

[2] Efron, Bradley and Gong, Gail. (1983) "A Leisurely Look at the Bootstrap, the Jackknife, and Cross-Validation." *The American Statistician*, 37(1), 36-48. Available at https://doi.org/10.2307/2685844

[3] Efron, Bradley and Tibshirani, Robert. (1986) "Bootstrap Methods for Standard Errors, Confidence Intervals, and Other Measures of Statistical Accuracy." *Statistical Science*, 1(1), 54-75. Available at https://doi.org/10.1214/ss/1177013817

[4] Efron, Bradley and Tibshirani, Robert. (1997) "Improvements on Cross-Validation: The .632 Bootstrap Method." *Journal of the American Statistical Association*, 92(438), 548-560. Available at https://doi.org/10.2307/2965703

[5] Efron, Bradley. (1979) "Bootstrap Methods: Another Look at the Jackknife." *The Annals of Statistics*, 7(1), 1-26. Available at https://doi.org/10.1007/978-1-4612-4380-9_41

[6] Efron, Bradley. (1983) "Estimating the Error Rate of a Prediction Rule: Improvement on Cross-Validation." *Journal of the American Statistical Association*, 78(382), 316-331. Available at https://doi.org/10.2307/2288636

[7] Efron, Bradley. (1986) "How Biased is the Apparent Error Rate of a Prediction Rule?" *Journal of the American Statistical Association*, 81(394), 461-470. Available at https://doi.org/10.2307/2289236

[8] Gong, Gail. (1986) "Cross-Validation, the Jackknife, and the Bootstrap: Excess Error Estimation in Forward Logistic Regression." *Journal of the American Statistical Association*, 81(393), 108-113. Available at https://doi.org/10.2307/2287975

[9] Harrell, Frank. (2015) *Regression Modeling Strategies: With Applications to Linear Models, Logistic Regression, and Survival Analysis, 2nd Edition*. New York: Springer.

[10] Harrell, Frank. (2018) "Comparison of Strategies for Validating Binary Logistic Regression Models." Date Accessed: 27FEB2020. Available at http://hbiostat.org/doc/simval.html

[11] Hastie, Trevor; Tibshirani, Robert; and Friedman, Jerome. (2017) *The Elements of Statistical Learning*. New York: Springer. Date Accessed: 27FEB2020. Available at https://web.stanford.edu/~hastie/ElemStatLearn/

[12] Molinaro, A.M.; Simon, R.; and Pfeiffer, R.M. (2005) "Prediction Error Estimation: A Comparison of Resampling Methods." *Bioinformatics*, 21(15), 3301-7. Available at https://doi.org/10.1093/bioinformatics/bti499

[13] Moons, K.G.; Altman, D.; Reitsma J.B.; Ionannidis, J.; Macaskill, P.; Steyerberg, E.W.; Vickers, A.J.; Ransohoff, D.F.; and Collins, G.S. (2015) "Transparent Reporting of a Multivariable Prediction Model for Individual Prognosis Or Diagnosis (TRIPOD): Explanation and Elaboration." *Annals of Internal Medicine*, 162(1), W1–W73. Available at https://doi.org/10.7326/M14-0698

[14] National Center for Health Statistics. "National Health and Nutrition Examination: Sample Code." Date Accessed: 27FEB2020. Available at https://wwwn.cdc.gov/nchs/nhanes/tutorials/samplecode.aspx

[15] Pyle, Lisa. (2007) "ODS, An Introduction to Creating Output Data Sets." *Proceedings of the NorthEast SAS Users Group 2007 Conference*, Baltimore, MD. Available at https://www.lexjansen.com/nesug/nesug07/cc/cc30.pdf

[16] SAS Institute. "ODS OUTPUT statement." *Output and Graphics: Output Delivery System*. Date Accessed: 27FEB2020. Available at https://documentation.sas.com/?cdcId=pgmsascdc&cdcVersion=9.4_3.4&docsetId=odsug&docsetTarget=p0oxrbinw6fjuwn1x23qam6dntyd.htm&locale=en

[17] SAS Institute. "ODS TRACE statement." *Output and Graphics: Output Delivery System*.

Date Accessed: 27FEB2020. Available at https://documentation.sas.com/?cdcId=pgmsascdc&cdcVersion=9.4_3.4&docsetId=odsug&docsetTarget=p1fpt3uuow90o3n155hs7bp1mo7f.htm&locale=en

[18] SAS Institute. "SAS® 9.4 DATA Step Statements: Reference." *DATA Step Programming*. Date Accessed: 27FEB2020. Available at https://documentation.sas.com/?cdcId=pgmsascdc&cdcVersion=9.4_3.4&docsetId=lestmtsref&docsetTarget=titlepage.htm&locale=en

[19] SAS Institute. "SAS® 9.4 SQL Procedure User's Guide, Fourth Edition." *Base SAS Procedures Guide*. Date Accessed: 27FEB2020. Available at https://documentation.sas.com/?cdcId=pgmsascdc&cdcVersion=9.4_3.4&docsetId=sqlproc&docsetTarget=titlepage.htm&locale=en

[20] SAS Institute. "Shared Concepts and Topics: CODE Statement." *SAS/STAT User's Guide*. Date Accessed: 27FEB2020. Available at https://documentation.sas.com/?cdcId=pgmsascdc&cdcVersion=9.4_3.4&docsetId=statug&docsetTarget=statug_introcom_sect013.htm&locale=en

[21] SAS Institute. "The GLM Procedure: CODE Statement." *SAS/STAT User's Guide*. Date Accessed: 27FEB2020. Available at https://documentation.sas.com/?cdcId=pgmsascdc&cdcVersion=9.4_3.4&docsetId=statug&docsetTarget=statug_glm_syntax05.htm&locale=en

[22] SAS Institute. "The GLMSELECT Procedure: MODEL Statement." *SAS/STAT User's Guide*. Date Accessed: 27FEB2020. Available at https://documentation.sas.com/?cdcId=pgmsascdc&cdcVersion=9.4_3.4&docsetId=statug&docsetTarget=statug_glmselect_syntax07.htm&locale=en#statug.glmselect.glscvmethod

[23] SAS Institute. "The HPLOGISTIC Procedure: PARTITION Statement." *SAS/STAT User's Guide: High Performance Procedures*. Date Accessed: 27FEB2020. Available at https://documentation.sas.com/?cdcId=pgmsascdc&cdcVersion=9.4_3.4&docsetId=stathpug&docsetTarget=stathpug_hplogistic_syntax11.htm&locale=en

[24] SAS Institute. "The LOGISTIC Procedure." *SAS/STAT User's Guide*. Date Accessed: 27FEB2020. Available at https://documentation.sas.com/?cdcId=pgmsascdc&cdcVersion=9.4_3.4&docsetId=statug&docsetTarget=statug_logistic_overview.htm&locale=en

[25] SAS Institute. "The PLM Procedure." *SAS/STAT User's Guide*. Date Accessed: 27FEB2020. Available at https://documentation.sas.com/?cdcId=pgmsascdc&cdcVersion=9.4_3.4&docsetId=statug&docsetTarget=statug_plm_syntax.htm&locale=en

[26] SAS Institute. "The SURVEYSELECT Procedure." *SAS/STAT User's Guide*. Date Accessed: 27FEB2020. Available at https://documentation.sas.com/?cdcId=pgmsascdc&cdcVersion=9.4_3.4&docsetId=statug&docsetTarget=statug_surveyselect_overview.htm&locale=en

[27] Schumacher, M.; Binder, H.; and Gerds, T. (2007) "Assessment of Survival Prediction Models Based on Microarray Data." *Bioinformatics*, 23(14), 1768-74. Available at https://doi.org/10.1093/bioinformatics/btm232.

[28] Simon, Jim. (2015) "Random Sampling: What's Efficient?" *SAS Learning Post*. Date

Accessed: 27FEB2020. Available at
https://blogs.sas.com/content/sastraining/2015/09/04/random-sampling-whats-efficient/

[29] Steyerber, E.W.; Bleeker, S.E.; Moll, H.A.; Grobbee, D.E.; and Moons, K.G. (2003) "Internal and External Validation of Predictive Models: A Simulation Study of Bias and Precision in Small Samples." *Journal of Clinical Epidemiology*, 56(5), 441-447. Available at https://doi.org/10.1016/S0895-4356(03)00047-7

[30] Steyerberg, Ewout. (2009) *Clinical Prediction Models: A Practical Approach to Development, Validation, and Updating*. New York: Springer.

[31] Taber's Online. (2017) "Hypertension." *Taber's Medical Dictionary, 23rd Edition*. F.A. Davis Company. Date Accessed: 27FEB2020. Available at www.tabers.com/tabersonline/view/Tabers-Dictionary/765649/all/hypertension

[32] Wicklin, Rick. (2012) "Simulation in SAS: The Slow Way or the BY Way." *The DO Loop*. Date Accessed: 27FEB2020. Available at https://blogs.sas.com/content/iml/2012/07/18/simulation-in-sas-the-slow-way-or-the-by-way.html

[33] Wicklin, Rick. (2013) "Turn Off ODS when Running Simulations in SAS." *The DO Loop*. Date Accessed: 27FEB2020. Available at https://blogs.sas.com/content/iml/2013/05/24/turn-off-ods-for-simulations.html

[34] Wicklin, Rick. (2014) "Sample with Replacement in SAS." *The DO Loop*. Date Accessed: 27FEB2020. Available at https://blogs.sas.com/content/iml/2014/01/29/sample-with-replacement-in-sas.html

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Please contact the authors as follows:

Isaiah Lankham
Senior Research Analyst
U. of California Office of the President
1111 Franklin Street
Oakland, CA 94607
Phone: +1-510-987-9776
E-mail: Isaiah.Lankham@ucop.edu

Matthew Slaughter
Statistical Research Analyst
Kaiser Permanente Center for Health Research
3800 N Interstate Avenue
Portland, OR 97227
Phone: +1-503-335-2400
E-mail: Matthew.T.Slaughter@kpchr.org