

Rice Analytics – Reduced Error Logistic Regression



MyRELR™ 2.03.3 Manual

Copyright © 2008-2011, Rice Analytics, St. Louis, MO (USA)

SECTION 1: REDUCED ERROR LOGISTIC REGRESSION (RELR) OVERVIEW

Throughout this manual, MyRELR™ refers to the branded software product for SAS users, whereas Reduced Error Logistic Regression (RELR) refers to the predictive modeling method. RELR is a relatively new predictive modeling method that can return models with significantly reduced error relative to standard methods. For example, an independent pilot report is that the lift in the validation sample K-S statistic may be as high as 25 points with the variable selection approach called ParsedRELR™ compared to standard decision tree and logistic regression methods with a large number of variables (Pruitt, 2009). RELR's reduced error effects are most pronounced with wider datasets, encompassing many variables in relation to observations. However, new independent research now suggests that RELR can also show significant classification accuracy improvement relative to other algorithms in taller datasets (Ball, 2011). RELR's predictive accuracy advantages are also seen both with averaged squared error or Brier Score (Rice, 2008). The Brier score will be extremely small when the predicted probabilities truly reflect the frequencies of the predicted event (Harrell, 2001). That is, a weather forecast with a 95% chance of snow may have the same classification accuracy as a forecast with a 51% chance, but clearly the weather forecast of 95% would be more accurate and useful if it does actually snow 95% of the time.

One reason for RELR's performance lift is because it has structures to model and reduce error. The result is that RELR is not subject to the 10:1 rule that requires 10 rarer target value observations for each linear independent variable in a binary logistic regression model. Nor is RELR subject to far greater Dahlquist restrictions on numbers of variables with nonlinear variables and interaction effects which require fewer than 2000 variables with 1,000,000 observations. In fact, troublesome overfitting may never happen with Parsed RELR variable selection. Also, RELR can effectively model many, many times more than the number of variables entered into a model, because it only models the variables that would be important if all variables had been included. This last effect is possible through RELR's variable shortlisting feature.

The RELR formulation is reviewed in Rice (2008). Although the new evidence is that RELR can outperform with taller datasets involving greater than 10,000 observations (Ball, 2011), RELR's biggest advantage will always be most apparent in difficult error-riddled problems such as those involving at least one of these conditions:

- High dimensional data with many possible independent variables and interactions
- Smaller sample sizes
- At least one large nonlinear or interaction effect that is hard to find and independent of simpler effects

To reiterate, RELR's biggest advantage seems to be that it returns highly accurate models with smaller samples sizes that are usually well below 10,000 observations. A much larger sample size may be needed for similar accuracy in other methods if there are a large number of independent variables. In many cases, even a larger sample size will not give standard methods the same accuracy as RELR. The reason is that standard methods cannot be run with larger sample sizes due to the curse of dimensionality if interactions are specified, whereas RELR can avoid the curse of dimensionality because it runs with very small sample sizes and because of its shortlisting feature. The effect of the error reduction is that RELR's accuracy can be very good at very small sample sizes. The important point is that there may be no need to run models with large numbers of observations in RELR, because RELR gives good accuracy at very small sample sizes. The fact that RELR gives accurate models with very small sample sizes can translate into enormous savings in data processing and data collection costs.

With a very small number of potentially important variables relative to the number of observations, RELR still may show comparable performance to Stepwise Logistic Regression when these important variables are easy to find as is the case with simple linear variables that do not involve interactions. Yet, with a high dimensional problem when important variables are more difficult to find, RELR will almost always have an advantage because it can select from a much more comprehensive set of variables than Stepwise Logistic Regression. RELR can handle large numbers of variables that include nonlinear and interaction effects in a reasonable period of time due to its ability to handle small sample sizes, due its batch processing ability, and because it only models the shortlist of most important variables. For example, if one took all two way interactions between 100 independent variables, the number of two way interaction variables would be $100^2/2$ or 5000 variables. If one also then formed nonlinear variables up to the 4th order polynomial from these interactions, the number of interaction variables could then be as high as 5000×4 or 20,000 variables, depending upon redundancy. This number of interaction and nonlinear variables is well within RELR's range, but obviously this would be well beyond the scope of Stepwise Logistic Regression. While nonlinear and interaction variables can also be modeled through black box approaches such as Neural Networks and Support Vector Machines, RELR may offer an advantage over these black box approaches. This is because one can see RELR's parameters just as easily as in Stepwise Logistic Regression and because these black box routines do not have RELR's error reducing capabilities. Finally, in contrast to Stepwise Logistic Regression, RELR's automatic variable selection method known as Parsed RELR does not contain any arbitrary "entry" and "stay" parameters. This automation is a further possible RELR advantage, but a user may still override Parsed RELR's automatically selected variables when desired.

Besides the usual assumptions in predictive modeling relating to stable measures that do not change across model building and implementation conditions, the primary assumptions required to employ this version 2.03 of MyRELR are similar to standard ordinal or binary logistic regression. A very basic RELR assumption is that all dependent variable observations are independent events and that one can specify independent variables that characterize any possible correlations or relationships across the independent events that form dependent variable observations. Logistic regression does not have the assumptions of normally distributed residuals, lack of spatial or serial correlation in residuals and lack of heteroskedasticity that plague least squares regression. A time series is an example where dependent variable observations are independent observations that also exhibit serial correlation, but one can specify variables such as seasonality and trend, along with lagging causal event effects, that characterize these serial correlations in the dependent variable such that any residual errors may not show serial correlation. A cross sectional political poll is an example where observations are independent observations that may exhibit spatial dependencies, but one can specify independent variables such as race or income or longitude or latitude that characterize these spatial correlation patterns such that residual errors may not show spatial dependencies. The problem that one needs to be aware about in logistic regression that is somewhat analogous to serial correlations of residuals in least squares is overdispersion, which will result from under specified models such as those lacking critical interaction or nonlinear effects, but RELR is unlikely to have this problem because it allows the inclusion of so many variables. A further assumption is that the dependent variable is a binary, ordinal, or interval category measured variable. For example, one will not be able to model a continuous numeric dependent variable, as one will need to form interval categories to model this using MyRELR as in a Forecasting application. Also, MyRELR does not allow the modeling of multi-class/multinomial dependent variables unless these are ordinal or interval category variables. The decision not to develop RELR for multinomial logistic regression other than with ordinal/interval category variables is based upon a desire not to have arbitrary parameters in regression models. Like standard multinomial logistic regression, RELR's multinomial logistic regression parameters would depend entirely upon the choice of the reference category. Such arbitrary parameters are very difficult to interpret when variable selection is allowed because the variable selection will depend upon the reference category. This problem is not seen in binary or interval/ordinal logistic regression. For that reason, we would urge users to break their multinomial categories into binary categories that can be modeled by separate binary logistic regression models, or if they are

ordered categories, to model them with ordinal RELR. Like standard ordinal regression, Ordinal RELR does require a “proportional odds” assumption, but this assumption may be fairly benign if the ordinal/interval categories have been formed from an underlying continuous dependent variable. An example of when the proportional odds assumption is not met is when the ordered category dependent variable is actually qualitatively different at different levels such as when smoking is both involved with increasing risk for a disease condition and also alleviates the symptoms of that disease in its most severe forms. The Ordinal RELR modeling approach follows SAS Proc Genmod and thus does not offer an automatic test for proportional odds, but the standard Score test for proportional odds is known not always to be accurate (Stokes, Davis and Koch, 2000). For this reason, we suggest that users validate RELR models with Ordinal/Interval Category dependent variables using face validity and validation sample accuracy procedures. If a model has reasonable face validity and validation sample accuracy, then it is likely that the proportional odds assumption is also reasonable in relation to those independent variables that are assessed as important. There is always a chance that Ordinal Regression will miss fundamental effects that show qualitative differences across the ordinal category, such as smoking’s effect on certain diseases such as diabetes.

While we do not offer true proportional hazards Survival Analysis, we recommend that users who wish to perform Survival Analysis may do so by categorizing survival time into interval categories and then use Ordinal RELR to build the survival model. Indeed, recent evidence indicates that Ordinal Regression may be at least as accurate as standard Survival Analysis modeling approaches in modeling survival time as a dependent variable (Gopal and Meher, 2008).

The choice of how one should form interval categories from continuous numeric data will influence the solution in RELR’s forecasting or survival analysis or other predictive model based upon interval category data. In cases where the original dependent variable is roughly normally distributed or at least symmetrically distributed, a relatively small number of quantiles can be formed that will not be likely to be much more accurate with increasing numbers of quantiles. If the original dependent variable is skewed and one or a few interval categories represent a disproportionate number of responses, then a simple one-dimensional cluster analysis of the dependent variable such as using K-means clustering may be helpful, where the number of clusters will not be as important as the fact that the clustering is sensitive to the response levels that are disproportionately represented. Note that while K-means is very much biased to favor equal sized clusters in higher dimensions, it also favors clustering which gives complete separation between clusters. In one dimensional clustering, it is always possible to get perfect separation. As a result, it is very easy to see that K-means will actually give highly imbalanced cluster sizes in one dimensional clustering in cases where such imbalance does make sense. The gap statistic can be used to find the number of K-means clusters in a solution that often makes very good sense in well separated clusters. Here is a link to a website that provides sample SAS code for computing the Gap Statistic and K-means clustering to produce the number of clusters:

<http://sas-x.com/2010/01/implementing-gap-statistic-for-clustering-number-estimation/>

In all cases involving interval category RELR, the number of levels will not be real important in determining accuracy if the RELR Predavg_[name of target variable] output variable is used for the prediction and if the prediction only requires accurate ranking of responses, so a small number of levels is preferred for parsimony reasons because each new level requires a new intercept. The RELR Predavg_[name of target variable] output variable is continuous in form and reflects the predicted weighted probability average of all interval categories, so it will be much less sensitive to the number of interval category levels than the other predicted output variables. However, for instances where interval accuracy is important such as in a forecast or survival analysis, then the accuracy of the clustering and number of clusters will have a big effect and the Gap Statistic will be useful to produce an objective way to get a number that will be useful in providing more accurate forecasts or survival analysis results.

One final note of caution is that MyRELR should not be applied to datasets where the missing values have been imputed with single imputation, as this violates the independence assumption. In cases with missing data, it is advisable to let MyRELR handle the imputation unless one's imputation is almost 100% accurate (probably through accurate business rules). Imputations that are at best good guesses have the effect of corrupting the RELR error model because they artificially reduce the standard error in the t ratio used in the error model, whereas RELR would compute these t-ratios prior to its own internal imputation to avoid this corruption.

SECTION 2: MyRELR INSTALLATION

This Windows installation for SAS 9.2 assumes that a directory exists entitled C:\Program Files\SAS. This directory and the entire MyRELR for SAS installation should be on the same hard disk where SAS is installed; if SAS is installed on a different hard disk, then you should use that drive letter for all library locations. For Enterprise Guide client-server installations, SAS is often installed on the server computer, so MyRELR also would need to be installed on the server computer in these cases. Other system requirements are in Appendix 1. Once one has established these basic requirements, the following three steps are required for the installation of MyRELR for SAS 9.2. The first two of these steps are identical for an Enterprise Guide 4.2 installation, but see Appendix 2 for a discussion of how to set up the library relrfors in Enterprise Guide:

- 1) Create Directory Location C:\SAS\relrforsas. With Windows Internet Explorer go to location C:\SAS and Create New Folder entitled relrforsas (case sensitive).
- 2) Copy the files sasmacr.sas7bcat and license.sas7bdat into the C:\SAS\relrforsas folder.
- 3) Enable **relrfors** library at SAS startup. To do this open SAS and perform the following three steps
 - A) From the View drop down menu that is seen in the SAS program editor, choose Explorer
 - B) From the left side panel that appears, right click on Libraries under SAS Environment, choose New.
 - C) In the 'New Library' dialog box that appears, type **relrfors** as the name of the library, keep the default setting for 'Engine' and check the 'Enable at startup' checkbox. Additionally, under 'Library information', use the 'Browse' button to set the location of this library to the directory location C:\SAS\relrforsas.

Also, information about MyRELR for other operating systems is available at Rice Analytics.

There are a number of warnings in documentation on SAS's website about how anti-virus software may interfere with SAS. Many of these problems are run time errors that concern a lock held on either SAS utility or SAS temporary *.data files by the anti-virus software when it performs a scan, so SAS cannot rename or copy such a file. Due to RELR's intense processing, it is especially susceptible to these problems. Thus, before running MyRELR, it is best to make sure that your anti-virus software excludes the following in its operations:

- the SAS.EXE process
- all *.sas7bdat and *.sas7bcat files
- the output directory where MyRELR writes output (in the sample RELR calling program below this is set through the LIBNAME statement and is under user control; That is, LIBNAME output 'C:\Documents and Settings\User\My Documents'; *output library location;

- the c:\sastemp "WORK" directory

Without these changes in your anti-virus software, many anti-virus software programs will cause SAS to generate the following error when it runs MyRELR:

ERROR: Rename of temporary member for WORK.[name of temporal file follows]

Such an error may occur during one run, but not happen in an identical run because the effect of the anti-virus software is intermittent. As an example of the above remedy, see SAS's Usage Note 19580 on McAfee's product, where SAS recommends putting SAS files and its temporary directory on the exclude list for anti-virus detection. Fortunately, MyRELR has been written to avoid such interference in SAS from McAfee and other anti-virus products, so MyRELR users may never see the kinds of problems reported on SAS's website. However, we cannot ensure that MyRELR avoids problems with newer anti-virus software not yet documented on SAS's website. For example, we know that the new freeware anti-virus software from Microsoft called Security Essentials™ can cause runtime errors in SAS, although such problems are not yet reported on SAS's website. If users see any evidence of a lock held on either SAS utility or SAS temporary *.data files so that SAS cannot copy or rename the file, we suggest that users search the SAS customer website for any new information about how their anti-virus software may interfere with SAS. We also suggest that users follow SAS's remedy for this problem if it occurs during any MyRELR run, which is probably similar to what we outline above in terms of those files and directories that should be excluded from anti-virus operations.

SECTION 3: RELRCALL SAMPLE PROGRAM

Users have been provided the RELRCALL.sas sample program, along with its input sas dataset – STOCKDATA. These are real data from a Nasdaq 100 traded stock – PTEN – over the course of almost 10 years. These variables include the binary target variable (dailychange) along with 6 numeric input variables (firvar, secvar, thivar, fouvar, fifvar, sixvar). Dailychange reflects the historical daily up-down behavior of this Nasdaq traded stock one trading day out from the measurement of the input variables. The input variables firvar-sixvar reflect short term technical indicators such as trend and volume change trends.

RELRCALL shows the main components of calling RELR. First of all, Libraries need to be defined. Secondly, Input and Output datasets should be defined. Next, Variable Definitions need to be input. Finally, Design and Speed Tuning parameters need to be given. After these components have been defined, RELR is called simply as a standard SAS macro call %relr. This RELRCALL.sas program is listed below and each of these components will be reviewed in subsequent sections.

```
options nosource nonotes;
/*-----
--+
|
|   Title : RELRCALL
|
|
|
|
|
|
|
```

```

|
| NOTES: Sample program showing how to call RELR Macro from SAS
+-----+
--*/
* ----- Libraries -----;
LIBNAME inputa 'C:\Documents and Settings\User\My Documents'; *input library
location;
LIBNAME output 'C:\Documents and Settings\User\My Documents'; *output library
location;
%let input = inputa;
%let output = output;
OPTIONS MSTORED SASMSTORE=relrfors; *relrfors library should be automatically
enabled at SAS startup;

* ----- Input and Output Dataset Names -----;
%let RELR_INPUT_DATA = stockdata; *raw input file containing all variables and
observations as SAS dataset;
%let RELR_OUTPUT_VALIDATE = RELR_OUTPUT_VALIDATE; * name of output validation
dataset;
%let RELR_OUTPUT_TRAIN = RELR_OUTPUT_TRAIN; * name of output train dataset;
%let RELR_OUTPUT_SCORE = RELR_OUTPUT_SCORE; * name of output score dataset;
%let RELR_OUTPUT_REPORT = RELR_OUTPUT_REPORT; * name of output report dataset
contain regression parameters;
%let RELR_OUTPUT_PERFORMANCE = RELR_OUTPUT_PERFORMANCE; *name of output dataset for
binary models that contains performance statistics across different probability
thresholds;
%let RELR_OUTPUT_FITSTATS = RELR_OUTPUT_FITSTATS; * name of ouput dataset that
contains fit statistics and performance at the K-S Statistic threshold;

* ----- Variable Definitions -----;
%let RELR_TARGET = dailychange; * name of target variable - there can be only one
target variable;
%let RELR_ID = ; *names of all id variables;
%let RELR_NONOMINAL_INDEPENDENT = firvar secvar thivar fouvar fifvar sixvar; * non-
nominal independent variables;
%let RELR_NOMINAL_INDEPENDENT = ; *names of all independent variables that are
nominal;
%let RELR_INCLUDE = ;
%let RELR_EXCLUDE = ;

* ----- Design and Speed Tuning -----;
%let RELR_TARGET_LEVEL = BINARY; * - must be BINARY, INTERVAL or ORDINAL;
%let RELR_INTERACTION_ORDER = 2; * must be either 1, 2 or 3;
%let RELR_MODE = POLYNOMIAL; * - must be either LINEAR or POLYNOMIAL;
%let RELR_SPLIT = .3419; * proportion of randomly selected training sample from
RELR_INPUT_DATA data;
%let RELR_SPLITV = .6581; * proportion of validation sample selected randomly from
remaining observations in RELR_INPUT_DATA after training sample has already been
selected - sum of RELR_SPLIT and RELR_SPLITV cannot be greater than 1;
%let RELR_METHOD = PARSED; * - must be PARSED or FULLBEST;

```

```

%let RELR_BATCHES = 0; * default of 0 runs batches of half the number of
independent variables for interactions and only one batch for no interactions -
users can override this by setting to explicit number of batches - this parameter
only affects the speed of processing and not the accuracy;
%let RELR_SHORTLIST_SIZE = 90; * must be greater than or equal to 30 or will be
reset if possible;
%let RELR_OVERSAMP=1; * this parameter setting only has an effect on an automatic
scoring run with a BINARY dependent variable and never has any effect on training;

```

```

%relr;

```

```

*The following system options have been affected and are reset here;
options MERROR SERROR MPRINT NOTES SOURCE SOURCE2 MTRACE DKRCOND=WARN
DKROCOND=WARN QUOTELENMAX NOFMterr;

```

```

*additionally RELR has reset the output to go to the output window prior to
terminating normally;

```

SECTION 4: LIBRARY DEFINITIONS

The **input** and **output** data libraries are defined with macro variables as shown above. The directory location of these input and output data libraries cannot be the location of the SAS work folder. The only potentially novel aspect of the library definitions is that the **relrfors** library needs to have been enabled at SAS startup (please see MyRELR installation instructions).

SECTION 5: INPUT AND OUTPUT DATASET DEFINITIONS

The input and output datasets are required to be SAS datasets. These datasets are defined for MyRELR through the macro variable names shown in the Let statements in the RELRCALL sample program. All names are required to be defined; users can use different names on the right hand side than are given above. These macro variables that determine the names are as follows:

```

RELR_INPUT_DATA - raw input file;
RELR_OUTPUT_VALIDATE - name of output validation dataset;
RELR_OUTPUT_TRAIN - name of output train dataset;
RELR_OUTPUT_SCORE - name of output score dataset;
RELR_OUTPUT_REPORT - name of output dataset for regression parameters;
RELR_OUTPUT_PERFORMANCE *name of output dataset for binary models that contains
performance statistics across different probability thresholds;
RELR_OUTPUT_FITSTATS - name of output dataset containing fit statistics;

```

The proportion of the input data that is randomly sampled into train and validation conditions is determined by the RELR_SPLIT and RELR_SPLITV parameters; these are completely independent samples. The RELR_OUTPUT_VALIDATE and RELR_OUTPUT_TRAIN datasets contain observations that form the validation and train conditions. Only observations with non-missing target variable values can go into train and validation conditions. These datasets also contain the originally named input variables that are used by the final model and the original non-standardized variable values, along with the output predictions. RELR_OUTPUT_SCORE contains all observations whether

missing or not. If a user wishes to score target variable values without knowledge of target variable values in a model building run, they should input these observations into the RELR_INPUT_DATA dataset and code the target variable as missing (a period ‘.’ is used for missing values of binary and numeric variables in SAS datasets). If users wish to score a target variable in a non-model building run, we offer two other options that are automatic scoring and open SAS code scoring (please see Section 11). Note that only numeric variables – binary, interval or ordinal - are allowed as the target variable. All such missing target value observations, along with all train and validation observations are output into the RELR_OUTPUT_SCORE dataset. The RELR_OUTPUT_SCORE contains those independent variables that are selected by the final model. The values of these independent variables in the RELR_OUTPUT_SCORE dataset are standardized and the variable names avoid MyRELR long variable names in accordance with names provided in the RELR_OUTPUT_REPORT file. *To summarize, RELR_OUTPUT_TRAIN and RELR_OUTPUT_VALIDATE can be used to see original input variable names and values, whereas RELR_OUTPUT_SCORE uses the names of variables that are used in the RELR_OUTPUT_REPORT file and shows standardized values of these variables. RELR_OUTPUT_REPORT renames these long variable names for readability if interactions are involved, so extremely long variable names involving interactions are not possible. We clearly show a list in the RELR_OUTPUT_REPORT file of how long variable names are renamed according to this more readable standard.*

RELR_OUTPUT_REPORT contains the regression report including the regression coefficients and tests of significance. RELR_OUTPUT_FITSTATS contains fit statistics such as average squared error for the train and validation conditions and the K-S Statistic for binary models. RELR_OUTPUT_PERFORMANCE is only generated for binary models. It gives detailed performance numbers involving hits, misses, correct rejections, and false alarms across the entire range of probability thresholds. It would be useful if somebody wanted to construct a Receiver-Operator-Curve based upon sensitivity (hit rates) and 1-specificity (false alarm rates).

SECTION 6: VARIABLE DEFINITIONS AND VARIABLE NAMING STANDARDS

The following macro variable definitions are used to define the four types of input variable definitions:

```
%let RELR_TARGET = dailychange; * name of target variable - there can be only one
target variable;
%let RELR_ID = ; *names of all id variables;
%let RELR_NONOMINAL_INDEPENDENT = firvar secvar thivar fouvar fifvar sixvar; * non-
nominal independent variables;
%let RELR_NOMINAL_INDEPENDENT = ; *names of all independent variables that are
nominal;
```

RELR_TARGET is the name of the target or dependent variable; there can only one target variable. RELR_ID is the list of all id variables; there can be more than one id variable. RELR_NONOMINAL_INDEPENDENT is the list of all independent variables that are not nominal, whereas RELR_NOMINAL_INDEPENDENT is the list of all independent variables that are nominal. The only variables that can be character variables are nominal and id variables. However, MyRELR automatically dummy codes nominal variables into binary variables. One can review the RELR_OUTPUT_TRAIN and RELR_OUTPUT_VALIDATE files to review how this dummy code produces new binary variables from the original nominal variables.

A user is allowed to use independent variable names that are longer than 8 characters provided that these correspond to SAS naming standards of 32 or fewer characters and no special characters other than the underscore ‘_’ character are employed. However, when MyRELR encounters an independent variable name that is longer than 8 characters, it automatically renames these input independent variables for its internal, RELR_OUTPUT_REPORT and RELR_OUTPUT_SCORE use. **RELR always writes out how it renames a variable to the RELR_OUTPUT_REPORT file.** Both the INPUTNAME – the

potentially longer name of the original variable including its nominal category - and REPORTSCORENAME – the shorter name used in reporting and scoring files - are given as fields in the RELR_OUTPUT_REPORT file on the right side of the ProbChiSq column. Labels are also given there if these variables have labels.

The total number of input independent variables after the recoding of nominal variables into binary variables cannot exceed 3550 variables, but this is before interactions, dummy coded missing value variables, and nonlinear variables. Thus, RELR can model many more times this number through interaction, polynomial, and dummy coded missing status variables that are computed internally. Thus, the effective number of variables that MyRELR can handle is only limited by the processing speed of the computer – it also allows batch processing and shortlist selection to speed this processing. For these reasons, it is generally a good idea to let MyRELR perform all interaction, nonlinear and missing status variable computations and only input those linear variables that one wishes to use as a basis for MyRELR internal interaction, polynomial, and missing status definitions.

When MyRELR computes polynomial, interaction, and missing status variables internally, it can form variable names for internal or reporting/scoring use that are longer than 8 characters. The following standards apply to how MyRELR names polynomial, interaction and missing status variables for reporting and scoring use in the datasets RELR_OUTPUT_REPORT and RELR_OUTPUT_SCORE:

TABLE 1: REPORT AND SCORE DATASET NAMING CONVENTIONS

| VARIABLE | RULE | EXAMPLES |
|-------------|---|--------------------------------------|
| INPUT | 8 or fewer characters | INCOME, DISTANCE, GENDER |
| INTERACTION | Concatenation with _ between components | INCOME_DISTANCE, INCOME_GENDER |
| POLYNOMIAL | The text p2, p3, or p4 is concatenated | INCOME_DISTANCEp2, INCOME_DISTANCEp3 |
| MISSING | The text mi is concatenated | INCOMEmi, DISTANCEmi |

In the above Table 1, a polynomial variable name that is quadratic would have ‘p2’ as its last two characters. A cubic variable name would have ‘p3’ as its last two characters. A quartic variable name would have ‘p4’ as its last two characters. Additionally, all original linear input variables that have missing values are dummy coded into missing status variables to reflect whether or not a value is missing in a given observation, so a missing status variable has ‘mi’ as its last two characters. Note that RELR’s interpretation of missing values is consistent with the SAS dataset definitions where a period “.” is used to indicate a missing numeric variable value and a blank “ ” is used to indicate a missing character variable value. Users should avoid using independent variable names as inputs that have p2, p3, p4, or mi as the last two characters to avoid confusion with MyRELR naming conventions. Additionally, users should avoid variable names which use the underscore “_” character as this will also cause confusion in RELR’s processing of variable names with interactions. A variable that is named INCOME_DISTANCEp3 would represent the following result (INCOME X DISTANCE)³, as the p3 indication is applied to the entire interaction. As another example, a variable that is named AGE_INCOMEp2 would represent the following result (AGE X INCOME)², as once again the p2 indication is applied to the entire interaction.

RELR automatically standardizes all independent variables to have a mean of 0 and a standard deviation of 1 based upon training sample data. This is all done automatically within RELR. If users wish to see the standardized value of a given

variable that is selected in the final MyRELR model, they can see this value for each and every observation that is output in the RELR_OUTPUT_SCORE dataset. This would include all training, validation, or scored observations.

If a user wishes to include or exclude specific variables in the PARSED method, the RELR_INCLUDE and RELR_EXCLUDE macro variables need to be defined with the names of those variables as they appear in the RELR_OUTPUT_REPORT dataset. RELR_INCLUDE and RELR_EXCLUDE names are not case-sensitive, but the limit on RELR_INCLUDE is ten variable names. RELR_INCLUDE and RELR_EXCLUDE only apply to variable selection through Parsed RELR models, as they do not apply to Fullbest RELR models (see below for a discussion of Parsed and Fullbest models).

An example usage of RELR_INCLUDE and RELR_EXCLUDE is shown below where firvar thivarp2 and sixvarp2 are variable names:

```
%let RELR_INCLUDE = firvar thivarp2;  
%let RELR_EXCLUDE = sixvarp2;
```

Note that one needs to give the precise REPORTSCORENAME that MyRELR uses in its RELR_OUPUT_REPORT file for this to work. Users can use the REPORTSCORENAME name of variables given there to quickly define variables in the RELR_INCLUDE and RELR_EXCLUDE statements. If you wish to define an interaction variable that MyRELR would use based upon these REPORTSCORENAME names, the simple rule is that that the interaction names are built from the reversed alphanumeric order. For example, an interaction variable built from variables named Z1 Z2 and X would be named Z2_Z1_X. Likewise, an interaction variable built from variables H1 and B would be named H1_B in RELR. If one wanted to also include or exclude a polynomial variable such as a second order polynomial built from these interactions, the names would be Z2_Z1_Xp2 or H1_Bp2. Through this simple logic, one can exclude or include any variable that MyRELR would use including interaction and polynomial variables.

SECTION 7: DESIGN AND SPEED PARAMETERS

Besides the above parameters that define the data structures, MyRELR has a very small number of additional input parameters under user control. These parameters determine factors related to the design of the MyRELR model and the speed with which it executes. None are case-sensitive and are shown in Table 2.

TABLE 2: RELR DESIGN AND SPEED MACRO VARIABLE PARAMETERS

| | |
|------------------------|--|
| RELR_TARGET_LEVEL | BINARY, INTERVAL, or ORDINAL |
| RELR_INTERACTION_ORDER | 1, 2, or 3 |
| RELR_MODE | LINEAR or POLYNOMIAL |
| RELR_SPLIT | >0 and <=1 |
| RELR_METHOD | FULLBEST or PARSED |
| RELR_BATCHES | >= 0 |
| RELR_SHORTLIST_SIZE | >= 30 for FULLBEST or >=90 for PARSED; (will be reset to be multiple of 30 if >30) |

RELR_TARGET_LEVEL: This determines how the target variable is interpreted. It can be only BINARY, INTERVAL, or ORDINAL. All target variables must be numeric and BINARY variables must be coded 0 and 1

for all non-missing values. An error message will be produced if target variable values do not conform to these restrictions. Also, continuous variables should be converted to interval categories and the mean or cluster centroid of the interval should represent the interval value in order for the t-statistic in the error model to be accurate. Ordinal variables will need no transformation if their values represent appropriate ranks.

RELX_INTERACTION_ORDER: The RELX_INTERACTION_ORDER determines the degree of the interaction that is allowed in the input variables in the model. These interaction choices corresponds to 1 (or no interaction), 2 (or two way interaction) or 3 (or three way interaction). This interaction order that is selected will apply to all independent variables.

RELX_MODE: The RELX_MODE is either LINEAR or POLYNOMIAL. A linear model is one that does not include any higher order nonlinear components in its models. A polynomial model is one that includes the modeling of higher order nonlinear components in addition to linear components. When POLYNOMIAL is selected, quadratic, cubic and quartic polynomial terms will be computed for numeric independent variables that are not binary. The reason that these are not computed for binary independent variables is that these nonlinear terms are completely redundant with the binary variable.

RELX_SPLIT: This determines the proportion of the non-missing RELX_INPUT_DATA dataset observations that go into the train condition. If this parameter is set to zero, then MyRELX automatically assumes that this is a scoring run and automatic scoring will be used where the file to be scored and the parameters are automatically determined by parameters stored in the far right hand fields of the RELX_OUTPUT_REPORT file in the output library. If this file does not exist, or if the RELX_OUTPUT_TRAIN file that it names does not exist in the output library, an error message is given and the run terminates.

RELX_VSPLIT: This determines the proportion of non-missing sampled data from RELX_INPUT_DATA that go into the valid condition. This sample is completely independent from the training sample. The sum of the RELX_SPLIT and RELX_VSPLIT parameters cannot be greater than 1.

RELX_BATCHES: RELX_BATCHES determines the number of batches that MyRELX uses to build its shortlist of variables with the largest magnitude t-values. With a large number of independent variables, MyRELX could easily need to compute millions of t-values to determine the shortlist variables. This process can be very time consuming even on fast computers. The process of batching can greatly speed this process. The SAS log gives the user a constant update of this batching process while the model is being run. In this way, users can determine how long each batch takes to run and approximately how long a model will take to finish. A default setting of 0 tells MyRELX to set the batches on its own. When MyRELX encounters this default setting, it does not employ batches if there are no two way or three way interactions. With two or three way interactions and the default setting of 0, MyRELX will compute the number of batches to be half of the number of input variables. To override this default, one can set this RELX_BATCHES parameter to any integer value greater than 0.

RELX_METHOD: RELX_METHOD determines the type of model that MyRELX runs. This can be either FULLBEST or PARSED. They are compared below in Figure 1. FULLBEST returns the solution that has the maximum log likelihood in the training sample when progressively fewer variables are selected according to t-value magnitude screening (see Rice, 2008). PARSED solutions are solutions that result from a backward parameter selection process based upon multivariate important criteria sensitive to redundant variables that is consecutively applied until a solution is found that is approximately optimal in the sense of having the fewest number of parameters along with the best fit. The PARSED solution maximizes the entire RELX log likelihood function that includes the portion devoted to the RELX error model. Generally, PARSEDRXL is the preferred final model, because it will usually be more interpretable and will be approximately as accurate as

FULLBEST given a minimal sample size. In general PARSED models will be more reliable and accurate as the RELR_SHORTLIST_SIZE parameter is increased until it is at a point where models are reliably the same with further increases. PARSED models cannot be run with the RELR_SHORTLIST_SIZE being less than 90; only FULLBEST models can be run with this setting below 90. Additionally, when the total number of non-redundant independent variables is less than 30, PARSED models may not be run and only FULLBEST models may be run.

FIGURE 1: A COMPARISON OF FULLBEST AND PARSED RELR

| Method | Advantages | Disadvantages |
|-----------------|---|---|
| Fullbest | Usually more accurate with very small sample sizes (e.g. fewer than 500 rarer binary target values) | Usually has number of redundant variables and can be severely biased to favor smoother lower ordered/ odd polynomial solutions with small sample sizes, so not interpretable as explanatory model. May overfit. |
| Parsed | Small number of variables and potentially interpretable as “explanatory model”. Favors parsimonious solutions. Very little overfitting. | May not be as accurate as Fullbest with very small sample sizes |

RELR_SHORTLIST_SIZE: The shortlist size determines the number of variables that are input into the logistic regression portion of the RELR model. RELR overcomes the curse of dimensionality by only inputting those variables that have the largest magnitude t-values that reflect the effect of the independent variable on the dependent variable. There is statistical justification for this shortlist process based on the idea that the RELR standardized beta coefficients are approximately proportional to t-value magnitudes with a relatively large number of important variables, so this shortlist method gives one those variables that would have the largest beta weights if all variables had been entered into a model. Interested readers are referred to Rice (2008) for further information. To some extent, the larger the shortlist size, the more accurate will be the RELR model, but there is a point at which a larger shortlist size will have no effect. With very small sample sizes of input data, some tendency for overfitting may be seen with too large of a shortlist size in FULLBEST models, especially as this shortlist size is greater than the number of observations. With larger sample sizes of input data, there likely will be no gain

beyond a certain shortlist size, so users should be careful so that they do not set the shortlist size to be too large, as it will take longer to run a model without any gain in accuracy. The largest that the shortlist size can be is 3550. In typical RELR runs, a RELR_SHORTLIST_SIZE of below 400-500 and maybe in the range of 100-200 often gives PARSED models that will be replicated when larger values of this parameter are set, which is consistent with an optimal parameter setting. PARSED models are very susceptible to too low of a setting of this parameter. For this reason, the lower limit of this parameter for PARSED models is 90.

RELR_OVERSAMP: The oversampling parameter has no effect on a model training run. It only has an effect on automatic scoring when there is a BINARY dependent variable. In RELR, all training samples are handled as if the training sample is a representative sample of the population. In BINARY models, one may wish to oversample the target or nontarget condition in the case of relatively rare values in one condition or the other. If one has actually oversampled the target value =1 for the training sample in a BINARY model, one should set this parameter to this oversampling ratio during automatic scoring. Through Automatic Scoring, this will automatically adjust the output probabilities and the BINARY scoring threshold in the file defined by the RELR_OUTPUT_SCORE macro parameter to account for this oversampling. For example, if one had included twice as many observations with the Target value=1 than 0 in the training sample than would be expected based upon the population, the RELR_OVERSAMP parameter should be set to 2 to take this into account. Once again, this parameter setting has no effect on a training run.

SECTION 8: RELRCALL SAMPLE OUTPUT

The following Table 3 shows the RELR_OUTPUT_REPORT dataset that is created when RELRCALL is run as shown with the STOCKDATA input file. This was a PARSED run. Although 84 total variables were created through the interaction and polynomial variables, PARSED only selects two variables in its optimal solution. The estimates in this table are the logit regression coefficients. Note that a positive logit coefficient implies a positive effect on probability and a negative coefficient implies a negative effect; this may be different from standards in some SAS programs where the polarity may be reversed. The Chi Square and confidence interval measures are based upon Wald approximations and can be less accurate with smaller sample sizes.

TABLE 3: RELR_OUTPUT_REPORT FROM BINARY STOCKDATA SAMPLE MODEL (RELR_SPLIT=.3419)

| parameter | DF | Estimate | StdErr | LowerWaldCL | UpperWaldCL | ChiSq | ProbChiSq |
|------------|----|----------|---------|-------------|-------------|-------|-----------|
| Intercept1 | 1 | 0.10406 | 0.07724 | -0.0473 | 0.2554 | 1.82 | 0.1779 |
| FIRVAR | 1 | -1.04618 | 0.11185 | -1.2654 | -0.8269 | 87.48 | <.0001 |
| SIXVAR | 1 | 0.83916 | 0.10818 | 0.6271 | 1.0512 | 60.17 | <.0001 |

RELR_OUTPUT_FITSTATS in Table 4 shows the Hit % and False Alarm % that are computed using the K-S Statistic Threshold that is .5 in this model. The K-S statistic is the maximal difference between the Hit % and False Alarm %; the Threshold is the point where this occurs. MyRELR breaks the probability output values into 99 different intervals to get the K-S Statistic and its corresponding Threshold. Note that the measures concerning Squared Errors were computed on the basis of the degree to which the prob1 variable in the output data matches the target variable, where the prob1 variable reflects that probability that the target variable is 1. Hit, misses, correct rejection, and false alarm data are provided for both conditions. The Brier Score or Average Squared Error across all observations is given. ROC AUC Estimate is based upon a

trapezoidal rule approximation to get the Receiver Operating Curve's area under the curve. The Reversed ROC AUC Estimate reverses the axes in the ROC curve and then computes the area under the curve. One can plot the ROC curve through the data provided in the RELR_OUTPUT_PERFORMANCE dataset. The Reversed ROC AUC is comparable to what would be found if one reversed the 1 and 0 binary values in the target variable. ***One should exercise extreme caution in interpreting AUC results, as recent evidence suggests that the AUC is significantly less precise than straight classification error at small sample sizes and a worse correlation of AUC estimated error to actual error is also seen in larger sample sizes (Hanczar et al. 2010).*** To compound this problem, the trapezoidal rule approximation is also especially less accurate with smaller sample sizes when very few values are available for the intervals in the AUC calculation.

One may wish to oversample the target or non-target condition in a training sample to achieve sample balancing when one of these target conditions is relatively rare. Measures of classification accuracy and the relative ranking of the probability estimates can be relatively uncorrupted by sample balancing if one obtains a representative sample of both target and non-target conditions in the balanced training sample. Additionally, only the intercept is largely affected by sample balancing, so the form of the model will remain similar whether the sample is balanced or not provided that there is a large enough representative sample of both target and non-target events. Thus, the benefit to sampling balancing is that one can get by with a much smaller training sample and still get roughly the same model. However, the average squared error that is reported in a RELR training model will not be the same with a balanced vs. unbalanced sample.. When oversampling is done, one should set the RELR_OVERSAMP parameter in an automatic scoring run to the ratio of this oversampling. In this way, the probabilities that are produced by automatic scoring will be correctly calibrated to the population.

Because Parsed RELR does not have arbitrary parameters, it will return the same model given the same training data no matter who runs the model provided that the shortlist size has been set large enough to be optimal. However, like all predictive modeling methods, Parsed RELR will return different models with different training sample sizes. Usually, the effect is that the same core variables that were selected with smaller sample sizes are retained in the model at larger sample sizes, but a relatively small number of new variables can also be added in at larger sample sizes. If more variables are added in the model at larger sample sizes, the Parsed RELR model usually becomes more accurate. However, this increase in accuracy with increasing sample size can be very slight, such that RELR accuracy often asymptotes at a very small sample size. The built-in Training Sample Size Calculator that is output to the log when the training sample size can be increased is helpful to determine how much one can increase the training sample size to get a nearly optimal larger training sample size. This is discussed in more detail in Section 11.

TABLE 4: RELR_OUTPUT_FITSTATS FROM BINARY STOCKDATA SAMPLE MODEL

| Fit_Statistic | Target_Variable | Train | Validation |
|----------------------------------|-----------------|----------|------------|
| Number of Observations | dailychange | 813 | 1437 |
| Avg. Squared Error - Brier Score | dailychange | 0.210924 | 0.214562 |
| ROC AUC Estimate | dailychange | 0.731554 | 0.721414 |
| Reversed ROC AUC Estimate | dailychange | 0.731533 | 0.719821 |
| Classification % Error | dailychange | 31.11931 | 33.27251 |
| Hit % | dailychange | 60.38647 | 57.38397 |
| Hits | dailychange | 250 | 408 |
| Misses | dailychange | 164 | 303 |
| False Alarm % | dailychange | 22.30576 | 24.51791 |

| | | | |
|--------------------|-------------|----------|----------|
| Correct Rejections | dailychange | 310 | 548 |
| False Alarms | dailychange | 89 | 178 |
| K-S Statistic | dailychange | 38.08071 | 32.86606 |
| Threshold | dailychange | 0.52 | 0.52 |

Table 5 shows the fit statistics output dataset when RELR_TARGET_LEVEL was set to INTERVAL. Note that the RELR_OUTPUT_REPORT dataset gives the same result as when RELR_TARGET_LEVEL was set to BINARY, but the RELR_OUTPUT_FITSTATS gives a different result. In the case of an INTERVAL or ORDINAL target variable, the only measures of error are Squared Error measures. Additionally, these Squared Error measures are based on the variable Pred_[name of target variable] or Pred_Dailychange in this case that is output in the train, validation, and score output datasets.

TABLE 5: RELR_OUTPUT_FITSTATS FROM INTERVAL STOCKDATA SAMPLE MODEL

| Target_Variable | Fit_Statistic | Train | Validation |
|-----------------|------------------------|-------|------------|
| dailychange | Number of Observations | 813 | 1437 |
| dailychange | Sum of Squared Errors | 260 | 495 |
| dailychange | Average Squared Error | 0.319 | 0.344 |

Pred_[name of target variable] is the predicted value of the target variable in the RELR_OUTPUT_SCORE, RELR_OUTPUT_VALIDATE, and RELR_OUTPUT_TRAIN datasets. If the user has requested BINARY as the RELR_TARGET_LEVEL, the RELR_OUTPUT_SCORE, RELR_OUTPUT_VALIDATE and RELR_OUTPUT_TRAIN datasets will contain a variable called prob1 which is the probability of the target variable being 1. If the user has selected ORDINAL or INTERVAL for the RELR_TARGET_LEVEL, these RELR_OUTPUT_SCORE, RELR_OUTPUT_VALIDATE and RELR_OUTPUT_TRAIN datasets will contain a variable called Predavg_[name of target variable] or Predavg_Dailychange, in this case, which is the average predicted target variable value given the predictive model. In addition, another output variable is available that is called Pred_[name of target variable]. The rule for forming the Pred_[name of target variable] predicted value depends upon whether the BINARY or INTERVAL/ORDINAL conditions are chosen as the RELR_TARGET_LEVEL. If BINARY is chosen, all observations with a prob1 value of greater than or equal to Threshold are output to a value of 1 for Pred_[name of target variable] ; all with a prob1 value of less than Threshold correspond to a value of 0, where the Threshold is the K-S Statistic threshold as shown in Table 4 and output in the RELR_OUTPUT_FITSTATS dataset. If INTERVAL/ORDINAL is chosen, Pred_[name of target variable] is scored using the Predavg_[name of target variable] variable according to cut points that approximate the proportion of each ordinal/interval level that is observed in the training dataset. In other words, if 10% of the original training sample target variable observations have values of 1, then cut points of the Predavg_[name of target variable] variable will be formed so that 10% of all training observations will also receive a Pred_[name of target variable] of 1. The reason that BINARY and INTERVAL models give different results in this case where the target variables and input data are the same is because different cut points/thresholds are used to score values of 1 and 0 and because different variables are used to compute Squared Error measures. That is, average squared error is based upon the degree to which the Pred_[name of target variable] variable deviates from the target variable in the case of interval/ordinal regression. In cases of interval/ordinal regression

with more than two levels, cumulative probability estimates are shown in the output datasets that correspond to the number of estimates allowed, which is one less than the number of interval/ordinal categories. These probability estimates are represented as prob1, prob2, ...up to the number of cumulative probability estimates.

SECTION 9: INTERPRETATION OF INTERACTIONS

The most straightforward way to interpret interactions is through visualization. We offer a 30 minute training session in the Training and Licensing page of our website called “Running and Using MyRELR: Modeling, Scoring and Visualizing Effects” which shows an example of how visualizing logit results from interactions with SAS Graph can help one interpret interactions. An example of this source code to use SAS Graph is supplied with the software, but any graphical software such as Excel or R could be used in this same way.

Interaction effects result from combinations of independent variables. These effects are different across the various sub-groups that result from the independent variables. For example, a significant interaction between two binary variables – Male and Republican – could reflect an effect that has a relatively high probability in Male/Republicans and Female/Non-Republicans vs. the other sub-groups in this interaction (Female Republicans, Male Non-Republicans). RELR builds interactions based upon products of standardized variables; it then re-standardizes the resulting product. One reason for using standardized variables is that the resulting form of the interaction variable does not depend upon how the variables that form the interactions have been coded.

MyRELR also forms nonlinear effects on interaction variables that could include variables that are entirely composed of binary variables. Such significant nonlinear interactions reflect nonlinear patterns of effects across the constituent variables that formed the interaction. For example, a significant p2 or quadratic effect could reflect an effect that is differentially related to one of the four conditions that is formed from the interaction of two binary variables. For example, if the two binary variables used to form an interaction were Democrat/Non-Democrat and Republican/Non-Republican, a significant p2 effect such as voting for a particular candidate may represent an effect specific to the people who are self-described as Non-Democrats and Non-Republicans. When Automatic Scoring is run (Section 11), the RELR_OUTPUT_SCORE file returns the logit (log odds) for each independent variable or interaction and each observation and is helpful to visualize interaction and nonlinear effects. Once again, we suggest that users visualize such interaction effects to help understand them and we offer an example of how to do this in the training page on our website.

SECTION 10: DATA PREPARATION: NOMINAL CODING, BINNING, OUTLIERS, AND IMPUTATION

It generally may not be beneficial to employ standard data preparation on MyRELR input data through methods such as nominal coding, binning, outlier processing, or imputation. The reason is that MyRELR has automatic structures in place to deal with all of these issues.

The coding of nominal variable values into binary codes is performed automatically by MyRELR for all variables given in the RELR_NOMINAL_INDEPENDENT macro variable input. The coding features within ProcTransreg are used for this purpose. A missing value in a nominal variable should be blank if it is a character variable or a period if it is a numeric variable. In these cases, Proc Transreg will recode all nominal values into binary (0 or 1) codes with a period to reflect the missing value. Users can view the RELR_OUTPUT_TRAIN and RELR_OUTPUT_VALIDATE files to see how their nominal variables have been coded into binary variables.

Binning to model a nonlinear effect through a set of binned linear effects is not recommended because MyRELR has the ability to model nonlinear effects when the POLYNOMIAL mode is chosen. Binning is believed to be a source of increased

error in predictive models because binning intervals are necessarily arbitrary. Therefore, we recommend the use of the POLYNOMIAL mode for the modeling of nonlinear effects rather than binning. However, we also understand that there may be special business rules and reasons that require a certain amount of binning.

Unlike least squares regression, logistic regression offers a natural way to deal with outliers because there are relatively few dependent variable values in logistic regression. Hence, any outlier in an independent variable will be averaged in with many other observations in the dependent variable category. For this reason, inflated independent variable effects due to outliers will be less likely to happen in logistic regression. We believe that it is unlikely that any special processing such as log transforms to deal with outliers will be necessary in MyRELR other than simple business rules to clean data that are obviously out of bound.

Additionally, MyRELR has automatic structures to deal with missing values in independent variables. When it encounters independent variables with missing values, it imputes these values with single imputation where the imputed value represents the expected value of the missing variable based upon the mean value of the non-missing variable values. It also forms a missing status dummy coded variable for each variable with missing values to model the potential of structure in missing information that is not related to values that are missing at random. Note that RELR's interpretation of missing values is consistent with the SAS dataset definitions where a period "." is used to indicate a missing numeric variable value and a blank " " is used to indicate a missing character variable value. The important aspect of RELR's single imputation is that it is done after t-values used in the error modeling are produced. If a user imputes the data prior to running RELR, the imputed values can inflate the t-values and this often gives quite inaccurate error modeling results. To counter this problem, users should not impute their data prior to running MyRELR unless they have either business rules or an imputation method that they know to be very accurate. In any event, when MyRELR sees that there are no missing values in independent variables, it always outputs the following warning message to the SAS log as a precaution:

```
WARNING: There are no missing data in any independent variables.  
WARNING: If this is because you previously imputed your data,  
WARNING: this may cause serious overfitting problems in RELR due to inflated t-values.  
WARNING: Unless you are extremely confident in the accuracy of your imputation,  
WARNING: it is strongly recommended that you let MyRELR handle missing data.
```

SECTION 11: BEST PRACTICES IN BUILDING A RELR MODEL AND SCORING

The basic rule of thumb in building a RELR model is to start with a very simple model that runs relatively quickly and then add complexity. The benefit to starting simple is that it will return a solution that a user can quickly check to ensure that data structures and variable names have been input correctly. For example, if a user mistakenly inputs the name of an interval variable into the list of nominal independent variables and if this variable has many levels, then this could create a large number of unwanted binary variables that would not only be wrong but would also cause MyRELR to run much longer due to the larger numbers of variables. Such problems can be avoided by first running a simple model with a very small sample size. Once a user has verified that the basic data that is input into MyRELR is correct, one can then move to more complex models that are more computationally-intensive, but also are more useful. For example, one may often wish to start with a LINEAR mode, with RELR_INTERACTION_ORDER=1, and with RELR_SHORTLIST_SIZE being 90 or 120 and with a relatively small training sample size because this will give a relatively fast result. Users may then wish to add complexity and increase the sample size to end up with inputs such as RELR_MODE=POLYNOMIAL, and RELR_INTERACTION_ORDER=2 or 3 in a model that levels off in accuracy with a greater sample size. If it is a PARSED model, one may also employ larger values of RELR_SHORTLIST SIZE to ensure that one has the same solution as with a parameter setting of 90

or 120. Note that the computational cost of too large of a shortlist size is actually very minor with small training sample sizes, so one may also eventually just set the shortlist size to a larger value such as between 200-400 as a starting point. The goal is to end up with a model where the accuracy and solution parameters are not affected by having too small of a sample size or too low of a shortlist size setting. Note that 3 way interactions have not been found often in business applications.

How large of a training sample size is needed for an accurate model? In a binary model, we do recommend that training sample size is chosen so that the rarer binary target group has at least 30-100 non-missing members for all important independent variables for FULLBEST RELR, but more would be needed for PARSED RELR. This minimum sample will help to ensure that the t-values that are the basis of the error model are reliable. In an interval/ordinal model, we recommend that the sample size is chosen so that the correlation between each independent variable and the dependent variable and the resulting t-test of this effect can be expected to be reliable. This may require a sample size of at least 100 non-missing members for all important independent variables using FULLBEST RELR, but more observations are needed for Parsed RELR. Across a number of datasets we have seen, a basic rule of thumb seems to be that PARSED RELR models may not asymptote in accuracy and begin to return highly stable solutions until the rarer binary target group has at least 500 observations. This rule of thumb is just a starting point in thinking about an accurate PARSED RELR model, but it may be useful for many users. In cases involving extremely correlated variables, a larger sample size should be chosen for PARSED RELR, such as at least greater than 3300 observations in the rarer binary target group. A recent study (Ball, 2011) suggests that stable PARSED RELR models in the sense of having reasonably stable variable selection and regression coefficients across independent samples may be seen with roughly 1650 observations in the rarer binary group with extremely high correlation in independent variables, but close to perfect stability was still not observed then. So, if a user wishes to have close to perfect stability such as would be needed for causal modeling, a larger sample in PARSED RELR may still be required with extremely high correlation in independent variables. For an interval/ordinal model, no such simple rule of thumb can be now given, but reliable and accurate models will still be seen at very small sample sizes.

This following is an example with the Stockdata dataset shows RELR regression coefficient in the Parsed type of model. An initial value of .1 selected as the RELR_SPLIT parameter; we also select the validation sample size by setting the RELR_VSPLIT to be .9. The PARSED BINARY model with RELR_INTERACTION_ORDER either set to 2 or 3 is shown in Table 6. Note that PARSED RELR almost always returns highly significant regression coefficients in all cases except the intercept, even at very small samples such as in this case. Such highly significant regression coefficients are not always seen with FULLBEST RELR.

TABLE 6: RELR_OUTPUT_REPORT FROM BINARY STOCKDATA SAMPLE MODEL (RELR_SPLIT=.1)

| parameter | DF | Estimate | StdErr | LowerWaldCL | UpperWaldCL | ChiSq | ProbChiSq |
|------------|----|----------|---------|-------------|-------------|-------|-----------|
| Intercept1 | 1 | 0.08761 | 0.14018 | -0.1871 | 0.3624 | 0.39 | 0.532 |
| FIRVAR | 1 | -0.90843 | 0.19621 | -1.293 | -0.5239 | 21.44 | <.0001 |
| SIXVAR | 1 | 0.61507 | 0.16795 | 0.2859 | 0.9443 | 13.41 | 0.0003 |

For this model with RELR_INTERACTION_ORDER=2, the SAS log file message at the end was the following:

NOTE: 0.1942330076 was the smallest probability of error in an independent variable.
 NOTE: A larger training sample size usually causes a decrease in this error measure,
 NOTE: but this value cannot be smaller than 1.2E-308.

The smallest probability of error value of 0.157 is the smallest probability of positive or negative error in an independent variable in the RELR error model. More accurate models will tend to occur as this value approaches zero, but this smallest value cannot be less than approximately $1.2E-308$. When models approach or achieve this lower limit value, it is a good indication that RELR may be beginning to asymptote in its accuracy. Notice that the model at a larger sample size shown in Table 3 with RELR_SPLIT=.3419 is very similar to the model that was computed when RELR_SPLIT=.1, but Parsed RELR model can in general change by either adding or deleting variables with more observations. The SAS log message was the following when RELR_SPLIT=.3419 and RELR_INTERACTION_ORDER=2:

```
NOTE: 0.0022676584 was the smallest probability of error in an independent variable.  
NOTE: A larger training sample size usually causes a decrease in this error measure,  
NOTE: but this value cannot be smaller than 1.2E-308.
```

Note that FULLBEST models may outperform PARSED RELR models simply because FULLBEST can select a very large number of variables in a model. Once PARSED RELR asymptotes in accuracy, this effect where we see more accurate FULLBEST models will usually be very small, so PARSED models will be greatly preferred because they have far fewer variables. However, if users are primarily concerned with accuracy and not concerned with interpreting a model and if they have limited sample sizes available, they will be comfortable with a FULLBEST model, as FULLBEST models can also be significantly more accurate than PARSED models with very small sample sizes where for example the rarer condition is a binary model has fewer than 400 observations. FULLBEST can show some tendency for overfitting when the shortlist size approaches the number of observations, but FULLBEST's overfitting problems will usually be far less than those seen in standard logistic regression. Overfitting usually will be observed to a lesser extent in PARSED models, but again any overfitting that is observed is usually far less than in standard logistic regression.

A problem with extremely small sample sizes is that complete separation or quasi-complete separation is more likely to occur in the logistic regression solution. Complete separation occurs when there is some combination of independent variables that give a perfectly accurate prediction such that there is 100% classification accuracy, whereas quasi-complete separation may have ties in some of the observations. In these scenarios, the probability of the target becomes 1 for some of the observations. In these cases, there is no unique maximum likelihood solution, as there is no maximum. When this happens, SAS presents a warning message that the convergence of the model is problematic. MyRELR deals with this problem related to complete or quasi-complete separation by presenting a warning message as follows:

```
WARNING: The model did not converge because there is no maximum likelihood solution  
WARNING: due to complete or quasi-complete separation.  
WARNING: RELR will continue and try to achieve a maximum likelihood solution across fewer variables.  
WARNING: To help get a maximum likelihood solution across the complete variable set,  
WARNING: you may wish to increase the training sample size and rerun the model.
```

When RELR encounters complete or quasi-complete separation, it is able to continue on with the model and drop a reasonable estimate of the least important variable in the set and therefore obtain a maximum likelihood solution in a smaller set if one exists. This is true for both PARSED and FULLBEST methods. In many cases, this maximum likelihood solution that is achieved across fewer variables may be somewhat similar to that achieved across the original larger variable set when the training sample size is increased. However, there are cases where there will be marked differences. One example with such marked differences would be if one had two independent variables that gave complete separation in the classification accuracy of a target variable. If these two independent variables were orthogonal and each accounted for approximately

equal proportions of variance in the target variable, then the maximum likelihood solution obtained with only one of these independent variables would be a much poorer fit than that obtained with both variables. Such cases as this with a much poorer fit in a reduced model would be more likely to occur with 10 or fewer variables. For this reason, PARSED models give the last solution where separation occurred as the best solution, if that solution had 10 or fewer variables. In this case, RELR gives the following warning:

```
WARNING: The model did not converge because there is no maximum likelihood solution
WARNING: due to complete or quasi-complete separation.
WARNING: RELR will continue and try to achieve a maximum likelihood solution across fewer variables.
WARNING: This happened with 10 or fewer variables,
WARNING: so PARSED RELR will report the last solution where this occurs as its best model.
MyRELR TIP: To avoid this issue, try increasing the training sample size.
MyRELR TIP: Then rerun the model and the model may still converge.
```

In any event, the best way to avoid problems related to complete or quasi-complete separation is to include a large enough sample size where some classification error is observed across all solutions. In this case, RELR will be able to get a maximum likelihood solution across the complete variable set.

Scoring target variable values from MyRELR models is possible in several different ways. First, all sampled records from the RELR_INPUT_DATA automatically will be scored by MyRELR after a model run. A model run automatically happens when the proportion of the original sample selected as training sample is greater than 0. This scoring after a model is built is output to the RELR_OUTPUT_SCORE file. Second, if one wishes not to build a model, but only wishes to score the proportion of RELR_VSPLIT records existing in a RELR_INPUT_DATA file with a previously built model, the RELR_SPLIT parameter should be set to 0 in the calling program such as relrcall.sas. This automatic scoring feature will cause MyRELR to look in the output library for the RELR_OUTPUT_REPORT and the RELR_OUTPUT_TRAIN files that are named in the calling program. Scoring will proceed automatically in this case with a proportion of the records equal to the RELR_VSPLIT setting, but all scored records will be output to the RELR_OUTPUT_SCORE file.

Below is an example of automatic scoring through the relrcall.sas program without a model build. Assume that one has recently built a model and saved the model parameters in a RELR_OUTPUT_REPORT sas dataset file called Exemplereport. Next, assume that Exemplereport is still located in the output library that is in the physical location of 'C:\Documents and Settings\User\My Documents'. **Additionally, assume that the training sample sas dataset from this model build is located in a RELR_OUTPUT_TRAIN sas dataset called Exampletrain in this same output library.** The following relrcall program shows how to implement automatic scoring to score all records in a sas dataset called stockdata that exists in the input library. The actual proportion of records is determined by the RELR_SPLITV parameter, which is set to 1 here. A final assumption is that this RELR_INPUT_DATA dataset that is called stockdata here has the same fields as the original dataset used to build the model. The output sas dataset that is scored is that which is defined through the RELR_OUTPUT_SCORE parameter name, which is scoredoutput in this case:

```
* ----- Libraries -----;
LIBNAME inputa 'C:\Documents and Settings\User\My Documents'; *input library
location;
LIBNAME output 'C:\Documents and Settings\User\My Documents'; *output library
location;
```

```

%let input = inputa;
%let output = output;
OPTIONS MSTORED SASMSTORE=relrfors; *relrfors library should be automatically
enabled at SAS startup;

* ----- Input and Output Dataset Names -----;
%let RELR_INPUT_DATA = stockdata; *raw input file containing all variables and
observations as SAS dataset;
%let RELR_OUTPUT_VALIDATE = ; * name of output validation dataset;
%let RELR_OUTPUT_TRAIN = ; * name of output train dataset;
%let RELR_OUTPUT_SCORE = Scoredoutput; * name of output score dataset;
%let RELR_OUTPUT_REPORT = Examplereport; * name of output report dataset contain
regression parameters;
%let RELR_OUTPUT_PERFORMANCE = ; *name of output dataset for binary models that
contains performance statistics across different probability thresholds;
%let RELR_OUTPUT_FITSTATS = ; * name of ouput dataset that contains fit statistics
and performance at the K-S Statistic threshold;

* ----- Variable Definitions -----;
%let RELR_TARGET = ; * name of target variable - there can be only one target
variable;
%let RELR_ID = ; *names of all id variables;
%let RELR_NONOMINAL_INDEPENDENT = ; * non-nominal independent variables;
%let RELR_NOMINAL_INDEPENDENT = ; *names of all independent variables that are
nominal;
%let RELR_INCLUDE = ;
%let RELR_EXCLUDE = ;

* ----- Design and Speed Tuning -----;
%let RELR_TARGET_LEVEL = ; * - must be BINARY, INTERVAL or ORDINAL;
%let RELR_INTERACTION_ORDER = ; * must be either 1, 2 or 3;
%let RELR_MODE = POLYNOMIAL; * - must be either LINEAR or POLYNOMIAL;
%let RELR_SPLIT = 0; * proportion of randomly selected training sample from
RELR_INPUT_DATA data;
%let RELR_SPLITV = 1.0; * proportion of validation sample selected randomly from
remaining observations in RELR_INPUT_DATA after training sample has already been
selected - sum of RELR_SPLIT and RELR_SPLITV cannot be greater than 1;
%let RELR_METHOD = ; * - must be PARSED or FULLBEST;
%let RELR_BATCHES = ; * default of 0 runs batches of half the number of independent
variables for interactions and only one batch for no interactions - users can
override this by setting to explicit number of batches - this parameter only
affects the speed of processing and not the accuracy;
%let RELR_SHORTLIST_SIZE = ; * must be greater than or equal to 30 or will be reset
if possible;
%let RELR_OVERSAMP=1; * this parameter setting only has an effect on an automatic
scoring run with a BINARY dependent variable and never has any effect on training;

%relr;

*The following system options have been affected and are reset here;

```

```
options MERROR SERROR MPRINT NOTES SOURCE SOURCE2 MTRACE DKRICOND=WARN
DKROCOND=WARN QUOTELENMAX NOFMterr;
```

```
*additionally RELR has reset the output to go to the output window prior to
terminating normally;
```

Notice that no parameters other than the library names, a RELR_INPUT_DATA name, a RELR_OUTPUT_REPORT name, the RELR_OUTPUT_VALIDATE name, the RELR_OUTPUT_SCORE, the RELR_SPLIT, and RELR_SPLITV parameters are even read by the program in an automatic scoring run. Thus, all parameters that are not read are set to blank here. Also, note that the RELR_OUTPUT_TRAIN filename is contained within the RELR_OUTPUT_REPORT file when it is output, so this parameter is not read through the relrcall.sas program. In contrast to the RELR_OUTPUT_SCORE file that is returned when a model is built, the RELR_OUTPUT_SCORE file during automatic scoring returns the logit for each non-intercept effect selected by the model and at each observation. To avoid confusion with the standardized variables that are returned during a model build, these logit variables have the suffix “lo” appended to their names. Graphical views such as available in 3-dimensional surface graphs in SAS Graph may be helpful here to visualize the distribution of logits or their related odds across observations, especially with interaction effects. A training webinar on the visualization of these effects through a program such as SAS Graph is available at the Rice Analytics website and sample SAS Graph code has been provided with the installation.

Finally, notice that the RELR_OVERSAMP parameter is set here to 1. This is actually the default value that will occur if RELR_OVERSAMP is not defined. Users can set this to any positive value, but it will only have an effect on this automatic scoring run with a binary model. More details about running and using RELR can be found by accessing this training webinar through the above link.

SECTION 12: RELR’S ERROR MODELING AND VARIABLE SELECTION VALIDITY

Users do not need an in-depth understanding of RELR’s error modeling because this process is entirely automated. Yet, those with an interest in the underlying theory behind RELR or those who desire to know the limitations of validating a model will find this section useful. RELR’s error modeling assumes that the log odds of expected error in an independent variable should be inversely proportional to the t-value that reflects the effect of the independent variable on the dependent variable, although under appropriate situations the actual modeled error can be quite different from this expected error (Rice, 2008). This inverse relationship with t would give extreme error values with small magnitude t-values, so we assume that this formulation is consistent with Extreme Value Type I error shown to exist in logistic regression (Marley’s original proof cited in Luce and Suppes, 1965, independent proof by McFadden, 1974). This choice of t in the RELR error model is not arbitrary, as the student t-distribution and the standardized logistic probability distribution are essentially the same distribution (George and Ojo, 1980). George and Ojo showed that when the standardized logit coefficient is multiplied by a scaling constant, these two probability distributions differ by less than .005% with a small number of degrees of freedom of less than 10 and this difference rapidly goes to zero with larger numbers of degrees of freedom. Another demonstration by Davidson (1980) showed that the difference between two extreme value distributed variables has a logistic probability distribution. We showed that when our two supposed RELR extreme value distributed variables – positive and negative error – have their expected error, their difference gives logit coefficients that are approximately proportional to t values (Rice, 2008). Hence, the choice of the log odds of expected extreme value error as being proportional to 1/t gives all known relationships between extreme value error variables, the logistic distribution, and the student t-distribution. Another choice for the error such as $1/t^2$ would not preserve these known relationships and instead give an anomalous result such as a proportionality relationship between logit coefficients and t^2 instead of t. This would be inconsistent with the Davidson (1980) result that the difference between two extreme value variables should have a logistic probability distribution. Thus,

the choice of $1/t$ in the RELR error model is not arbitrary, although it is only an extremely good approximation given that the student t-distribution and the logistic probability distribution are only approximately equivalent given more than a handful of observations. This necessary relationship to t is empirically supported by many tests of RELR by us and our users over the years (please see Appendix 4 for a further discussion of t in relationship to the best practice of sample balancing).

FULLBEST models are the best full model according to likelihood criteria when variables are dropped according to the magnitude of the t values that reflect an effect on the dependent variable. RELR's full model standardized logit coefficients are roughly proportional to t -values across independent variables when there are a large number of variables and too small of a sample size to discriminate the importance of these variables. The reason for this result is because the difference between positive and negative error for each variable is roughly zero with many variables relative to the sample size (Rice, 2008). When the difference between estimated positive and negative error is approximately zero, the algebraic relationships within RELR dictate that there will be almost a perfect proportionality between logit coefficients and t -values (see Rice, 2008). With larger sample sizes, the perfect correlation between the logit coefficients and t values breaks down in full RELR models. However, we can always imagine that the variables that we employ are just a subset of the possible variables especially if very high order interactions and nonlinear effects were considered. With much higher dimensionality, there will again be many variables in relation to even a larger sample size and the proportional relationship between t -values and logit coefficients would be expected. Hence, the magnitude of t -values can be used generally to screen the importance of variables whether based upon a large or small sample, because we expect that the proportionality between t -values and logit coefficients will always hold if all possible candidate variables had been employed. Hence, the magnitude of the t -values at both small and large sample sizes can be used to screen the subset of variables that have larger magnitude logit coefficients and this is the basis of RELR's dimensionality reduction. That is, all PARSED models simply start out as full models and then backward variable selection is applied. When we know the rank ordering of the magnitude of the regression coefficients in the initial full model prior to backward variable selection, we can start with a full model with far fewer variables. This dimensionality reduction using the SHORTLIST_SIZE parameter critically assumes that suppressor effects are improbable. More precisely, the parameterization of the expected error as being proportional to $1/t$ is akin to assuming that suppressor variables are highly improbable and even impossible (see last two paragraphs of this section). Altogether, this is what makes such shortlisting based upon the magnitude of t values possible. Because of the shortlisting capability, one does not have to build a model based upon for example 80,000 variables. Instead one can build a model based upon a small subset such as 400 variables and achieve the same model through backward variable selection in PARSED RELR as with more variables. When one shortlists, there is always the danger that the shortlist is too short. In this case, the PARSED RELR solution will be a local maximum in the optimization. One will only know if the shortlist is too short by setting the shortlist parameter to a higher value, running a model and determining if the model has changed. However, practical experience will show that in most business cases a shortlist parameter setting of roughly 400 will work very well and not change with higher parameter settings. In the worst case, if the solution is not a global maximum, it will still be a high quality local maximum.

As seen in full models and often seen in FULLBEST RELR models with very small sample sizes, proportionality between β coefficients and t values is precisely what Naïve Bayes returns in its solutions. Yet, Naïve Bayes does not allow the modeling of interactions and Naïve Bayes forces this rigid proportionality between logit coefficients and t -values generally, as for example even when there are a small number of important variables that do not provide a good fit, whereas RELR does not have these limitations. In any event, given these partial parallels between RELR and Naïve Bayes, it may not be surprising that RELR also performs well with error-riddled problems such as those involving small sample sizes because Naïve Bayes is well known to perform very well in these same situations (Mitchell, 2005). However, the fact that RELR is not rigidly tied to this proportionality relationship with larger sample sizes and because RELR models interactions also allows RELR to be significantly more accurate than Naïve Bayes at either smaller or larger training sample sizes. While FULLBEST is closely

related to Naïve Bayes under special circumstances, FULLBEST will often not keep all variables in a model because its solutions would be less than accurate if that were done, but Naïve Bayes would keep all variables in a less accurate model. Figure 2 summarizes the relationships between RELR and Naïve Bayes. Note that PARSED RELR's big advantage compared to FULLBEST RELR is that the standardized β coefficients are sensitive to covariance or redundancy between variables and are not even remotely a function of univariate importance (t values). Thus, PARSED models are much more parsimonious, whereas FULLBEST models can keep a lot of redundant variables.

FIGURE 2: COMPARISON OF RELR AND NAÏVE BAYES

| Method | Relationship Between standardized β and t | Interactions |
|---|---|----------------|
| Naïve Bayes (Mitchell, 2005) | $\beta_r \propto t_r$ | Does not model |
| RELR/Many IVs with large magnitude t-values (Often Fullbest RELR) | $\beta_r \propto t_r$ (approximately) | Does model |
| RELR/Few Important IVs (Always Parsed RELR; Sometimes Fullbest RELR) | β_r also depends upon covariance with other IVs | Does model |

A further two RELR error modeling assumptions concern the symmetrical error constraints that 1)force the average probability of positive and negative error to be equal across variables and 2)also force there to be no bias in the average probability of positive vs. negative error across the even vs. odd polynomial variables (Rice, 2008). The effect of these two constraints is to assure that the average probability of positive and negative error is equal across odd polynomial variables and also across even polynomial variables. These constraints have progressively less weight on the solution as fewer and fewer variables are selected in a model. This second symmetrical error constraint is obviously only in place when RELR_MODE = POLYNOMIAL. There is always very high correlation between each linear variable and its corresponding cubic variable; there also is always very high correlation between each quadratic variable and its corresponding quartic variable for these standardized variables. These correlation groupings are the reason for grouping the odd polynomial variables - linear and cubic variables in one group and the even polynomial variables - quadratic and quartic variables in a second group in the error model.

The second constraint ensures that this equality of the average probability of positive and negative error is obeyed in both groupings. Without this second constraint, the regression coefficients and the variables selected will be much more random across independent samples and the solutions and the measures of model accuracy will be no better and often significantly worse. In FULLBEST RELR models with large numbers of variables and relatively small sample sizes, these constraints can often place much more weight on odd polynomial effects. Odd polynomials have lower spectral frequency content than even

polynomials. Because of this, a low pass filtering mechanism should give greater weight to odd polynomial effects than even polynomial effects given an equivalent fit to the data. We do not directly penalize even polynomials more than odd polynomials, but we do force the average probability of error to be same across these conditions. Our simulations also suggest that an assumption of error inversely related to t-values is akin to a low pass smoothing filter that selectively allows lower polynomial order/lower frequency components into the model. As a whole, the RELR error model with very small sample sizes usually has the effect of a low pass smoothing filter with many redundant variables as with FULLBEST models.

FIGURE 3: THE EFFECT OF SMOOTHING

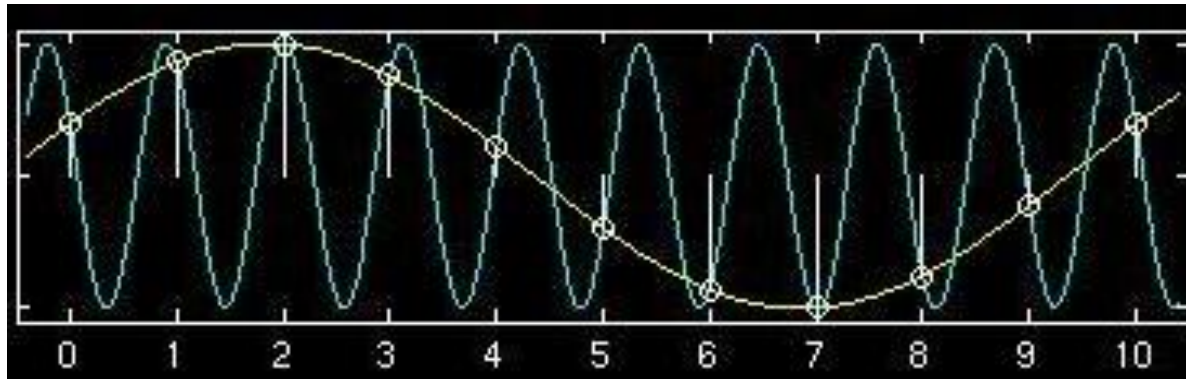


Figure 3 demonstrates the effect of smoothing by showing two of the infinite number of curves that fit a set of sample observations equivalently. Even though both curves fit the data points equally, an interpolation based upon smoothing that gives more weight to the lower frequency/lower order polynomial curve will be closer to the original points and will be more probable as tested by out-of-sample fit. Smoothing selectively removes higher frequency components that have a greater number of turning points; it is a form of low pass filtering. The smoothest possible effect is a simple linear effect, so simple linear variables will characterize solutions that favor smoother effects and variables. Smoothing is well known to produce less overfitting in predictive modeling. This explanation for why smoothing works is related to the well known Runge interpolation phenomenon that is the usual explanation for overfitting, as out-of-sample prediction is a form of interpolation. Runge's demonstration was that error in polynomial interpolation increases with higher order polynomial curves that have many turning points, so one interpretation of Runge's work is that the 'most probable' interpolation or predictive model is that which is weighted more with lower order and smoother polynomial curves. The fact that RELR tends to return such smooth solutions with small sample sizes is consistent with this interpretation. With larger sample sizes, a bias to avoid even polynomial variables with more turning points is not evident in RELR. Additionally, this bias to avoid even polynomial effects is always less evident in PARSED RELR than FULLBEST RELR even with small samples. One can see from Figure 3 that solutions such as those involving a greater number of turning points and therefore a greater number of even polynomial effects would be easier to resolve with a greater number of observational sampling points. Hence, RELR's abilities to find smoother solutions at smaller sample sizes and less smooth solutions at larger sample sizes are consistent with a maximum probability interpretation.

PARSED RELR often tends to avoid interaction effects; this is especially apparent with smaller training sample sizes. When two or three way interactions are formed, there are many more possible interaction variables than simple variables, but PARSED RELR often returns solutions that are biased toward simpler variables that are not interaction variables. When

compared to a simple linear effect based upon one of the original variables used to form the interaction, interaction effects might also be characterized as relatively unsmooth. However, this effect may also be connected to PARSED RELR's tendency to find parsimonious solutions, because it is not as apparent in FULLBEST models.

Along with simpler independent variables such as linear effects, parsimony is another important way to regularize a model and ensure that it avoids overfitting. The validity of parsimony as a guiding principle is well known, as a major component of valid models of real world phenomena in scientific problems is they tend to involve as few parameters as possible. The PARSED RELR maximum likelihood objective naturally returns parsimonious solutions. Also, PARSED RELR does not include arbitrary terms to deal with parsimony, as instead there are only observational and error probabilities in the RELR likelihood function and parsimony is a by-product of the maximum of this function (Rice, 2009). The RELR entropy/likelihood function can be thought of as a measure of the joint probability across all events that are in the model, whether they are from observational or error sources. The PARSED RELR maximum entropy/likelihood solution exists at the maximal value of this joint probability across all variable sets considered in the backward selection path. PARSED RELR maximizes this entropy/likelihood function by successively dropping an estimate of the least important variable and then refitting the model. Thus, PARSED RELR's probability model is not dependent upon a modeler's choice to use either AIC or BIC or any similar arbitrary device to introduce parsimony.

It would be ideal to have simulations that would be able to test the extent to which PARSED RELR solutions map to actual solutions, but we have not done this except in simple cases with one or two variables where perfect fitting solutions are known. In fact, our simulations have largely been concerned with error measures and our validations have largely been concerned with tests of face validity and split sample reliability. We do not believe that validation simulations to determine closeness of PARSED RELR solutions to simulated solutions are always useful. In order to perform a meaningful simulation to validate a model, one needs to have an understanding of the underlying probability distribution for the possible solutions that a model can produce. There are an infinite number of possible solutions even in simple predictive modeling problems with one variable – the standard maximum entropy/likelihood logistic regression solution just happens to be the most probable solution under the assumptions of the model. This complexity is magnified with high dimensional data because there are an infinite number of possible solutions in each of a large number of possible variable sets and this is a typical case in business applications. It would be very easy to construct a simulation that gives an actual solution that PARSED RELR does not find as its solution, but this would not explain much because we would not know the probability that this solution could occur in the real world. At the end, any simulation that uses the RELR entropy/likelihood function as its probability model would be biased to find PARSED RELR solutions as most probable solutions. This would be circular. This dependence of simulations on a probability model is ultimately why meaningful simulations may not be easy to perform. It may be that one is limited to face validity, split sample reliability/consistency, and if possible experimental verification in attempts to test the causal validity of PARSED RELR models with unknown solutions. Alternatively, if we know an actual real world causal solution to what can be formulated as a predictive modeling problem, we can test the validity of PARSED RELR's variable selection. In many cases, it will never be possible to interpret the independent variables as actual causal variables and instead will, at best, be able to interpret these PARSED RELR effects as "most probable" causal variables. , Even with only have "most probable" causal variables, there still always should be an interpretation as "explanatory variables" that are at least associated with the dependent variable in a predictive correlation. This "most probable" causal solution in PARSED RELR is a solution that requires as few measurements as possible and is reasonably consistent across independent samples and has face validity in terms of the signs of the coefficients. In many business predictive models, there may be an infinite number of solutions that all give close to the same accuracy performance, but the vast majority of these solutions may be extremely difficult to interpret due to a large number of variables or because the solutions are completely

unstable across independent samples. PARSED RELR avoids these problems involving a large number of variables, as it is the solution with the small set of most important variables and is quite stable compared to Stepwise Logistic Regression.

The “bias-variance dilemma” has been characterized as a limiting factor in regression modeling and machine learning with high dimensional data that will forever preclude the possibility of a general and automated most probable solution (Spall, 2003). As suggested by Figure 3 above, given the same fit to the training data, simple smooth curves are more likely to generalize better to validation samples than curves with more turning points. The dilemma is that this tendency to favor certain types of smoother polynomials amounts to a bias in solutions. Also, we do not know the actual probability for a solution *a priori*, so it has been argued that there is no way to find the most probable solution amongst the infinite number of possible solutions – some of which may be entirely composed of simple linear functions, and some of which may be composed of complex even polynomials or interaction effects. A compromise to the bias-variance dilemma is to have a human observer make a subjective judgment about how much smoothing is required to balance bias and variance. This subjective judgment also requires many other subjective judgments such as whether to use Ridge or LASSO penalized regression, or hybrid regularization such as Elastic Net amongst other possibilities. Or, whether to use to use AIC, BIC or other possibilities when parsimony is forced into the solution in the form of Stepwise regression. Cross-validation is then often employed to determine how well these subjective decisions have worked in producing an accurate model with out-of-sample data. Unfortunately, the end result is hardly a “machine learning” solution, but is instead heavily weighted by subjective human decisions and observations.

In contrast to a belief in inherent limitations caused by the bias-variance dilemma, RELR is based upon a belief that it is possible to get a most probable solution in spite of the bias-variance problems. Yet, RELR does not require that one has complete *a priori* knowledge of the probability distribution that governs different solutions. Instead, RELR requires assumptions about the error. These assumptions are that the error for a given variable is inversely related to the corresponding t value such that it is consistent with Extreme Value error and that the error is equally likely to be positive and negative without bias across odd and even polynomials. There is also an implicit assumption when one interprets a PARSED RELR model that the error measures are stable enough such that independent samples of observations would produce PARSED RELR models that are roughly similar. RELR also assumes that all target variable events across observations and all error events across independent variables are independent. With these assumptions and the assumption that the shortlist size has been set to a large enough value to prevent a local maximum, the PARSED RELR solution that exists at the maximum of the RELR likelihood function is the best estimate for the most probable solution for a given training sample of data. This is because this solution gives the maximum joint probability of all observational and error events in the model. The bias toward smooth solutions that is seen in the RELR models at smaller sample sizes does diminish with larger sample sizes. Because of this, bias-variance problems may be minimal in RELR given a large enough training sample size.

One interpretation of PARSED RELR’s optimization procedure may be that it is a form of graduated optimization. The graduated optimization procedure requires certain assumptions to ensure that it finds a global optimum (Blake and Zimmerman, 1987). Since the maximum likelihood solution with each set of variables is a globally optimal solution, PARSED RELR does return a globally optimal solution for each set of variables at each step in its backward selection process. This meets the first assumption of graduated optimization which is that the first optimization problem in the sequence is has a unique global optimum. Also, PARSED RELR’s final optimal solution does have the properties of the interesting problem that we wish to solve; it is the maximum of the maximum likelihood solutions in the backward selection path and is at least consistent with the global optimum. However, a third and very important assumption is that each solution in the PARSED RELR backward selection sequence is a point that resides within the region corresponding to the global optimum for the next step in the backward selection sequence. Said more simply, if we do not include a candidate solution

that is the globally optimal solution, then it is obvious that we will not find an optimum. Hence, if we have chosen our shortlist size to be too small and zeroed out critical variables, then we will fail to find the globally optimal solution across all variables that we measure. Even when we have included a large enough shortlist size, there is no test to ensure that we have achieved a global optimum across all variable sets. The only semi-objective test is to raise the shortlist size to a larger value and determine that this has no effect on the solution. When one starts with tens of thousands of variables in typical business data, we have seen that it appears unlikely that a larger shortlist size than 500-1000 has any effect on the PARSED RELR solution.

Our simulations show that PARSED RELR does not find solutions that involve suppressor variables even when all variables involved in the simulated suppression are included in the shortlist. However, this may be a major advantage. Suppressor effects involving regression coefficients with opposite signs to pairwise correlation effects are often evidence of multicollinearity error problems in regression modeling, as these effects often do not replicate in independent samples of observations. Suppressor effects are also very difficult to interpret and are termed “befuddlers” in the statistical learning literature (Auslander, 2009). While we cannot prove that PARSED RELR will always obtain regression coefficients with the same sign as the correlation, certain aspects of PARSED RELR would strongly prevent the likelihood of suppressor effects in solutions. For instance, if we have a suppressor effect with regression coefficients from two very highly correlated variables that are opposite in sign, then the more parsimonious solution will be a solution that involves just one of these variables or a variable that merges these two variables. Also, because one of the variables involved in the suppressor effect often has a near zero or a very low correlation to the dependent variable, this variable and its associated suppressor effect may not be included in the PARSED RELR shortlist and solution. Even if the shortlist were widened to include variables with very low correlations to the dependent variable, the RELR error model expects that error is inversely related to t value effects across variables. This severely penalizes variables with relatively low correlation to the dependent variable and effectively avoids suppressor effects in solutions. In many cases, a solution involving a suppressor variable will simply be infeasible because it requires that one or both of the offsetting regression coefficients have a larger magnitude than allowed in the RELR error model. Such regression coefficient scaling is determined by an error model that assumes that the total error cannot sum to be more than the sum of the individual positive and negative error magnitudes across all variables. For all of these reasons, the resolution of suppressor effects will be either highly improbable or infeasible in PARSED RELR solutions. This has the advantage of yielding solutions that have good face validity in RELR in the sense that the regression coefficients have signs that would be expected based upon correlation effects. Additionally, there is no experimental evidence that we are aware of that shows that suppressor effects are actually real world causal effects in biomedical, business, social or behavioral data. Even in theoretical statistical thermodynamics, a suppressor effect where all molecules exerting a force in one direction would be opposed by opposite molecule forces would be extremely unlikely. Recent evidence (Ball, 2011) highlights the problem with suppressor effects in method like stepwise regression. This study showed that stepwise regression selected between 2-3.5 times the number of variables as PARSED RELR and stepwise also had about 20% regression coefficients with wrong signs, whereas PARSED RELR had none in this study as in all studies ever done to date. Even though Stepwise selected so many more variables, RELR had a significantly better classification accuracy performance by as much as 3% in the Ball study. This highlights the idea that suppressor variables are likely to be an artefact due to error and RELR avoids these effects through its error modeling.

Thus, a final implicit assumption in the RELR error model and its PARSED RELR variable selection is that any solution that contains suppressor effects is either highly unlikely or impossible. This lack of suppressor effects in PARSED RELR is consistent with with no known experimental evidence to support the existence of suppressor causal effects in economics, business, biology, sociology, psychology and many other areas where predictive modeling is applied. This ability to avoid

suppressor effects is the reason that PARSED RELR gives models with regression coefficients that have interpretable signs consistent with causal effects.

REFERENCES

- Ammor, O., Lachkar, A., Slaoui, K., Rais, N. (2009). Assessing the quality of fuzzy partitions in overlapping datasets using maximum entropy principle. *International Journal on Computer Science and Information Systems*, Vol. 4, No. 1 pp. 75-84.
- Auslender, L. (2009). Suppression, enhancement, coefficient interpretation, and co-linearity in the linear regression model. Presentation given at INFORMS Conference, New York City.
- Ball, T. (2011). Modeling first year college achievement using RELR. Independent research report available at <http://www.riceanalytics.com/wsn/page13.html>.
- Blake A. and Zisserman A., (1987) *Visual Reconstruction*, MIT Press.
- Davidson, R.R. (1980). Some properties of families of generalized logistic distributions, *Statistical Climatology, Developments in Atmospheric Science*, 13, S. Ikedia et al. (ed.), Elsevier, New York.
- George, E.O. and Ojo, M.O. (1980). On a generalization of the logistic distribution. *Annals of the Institute of Statistical Mathematics*, Vol. 32, No. 2, A, pp. 161-169.
- Gopal, R.K and Meher, S.K. (2008) Customer churn time prediction in mobile telecommunication industry using ordinal regression. Book chapter in *Advances in Knowledge Discovery and Data Mining*, Springer: Berlin-Heidelberg.
- Hanczar, B., Hua, J., Sima, C., Weinstein, J., Bittner, M. and Dougherty, E.R. (2010). Small-sample precision of ROC-related estimates. *Bioinformatics* 26 (6): 822-830.
- Harrell, F. (2001). *Regression Modeling Strategies*, New York, Springer-Verlag.
- Khosla, D., Singh, M., and Rice, D. (1995). Three dimensional EEG source imaging via the maximum entropy method. *IEEE Nuclear Science Symposium and Medical Imaging Record*. Vol. 3, pp. 1515-1519.
- Luce, R.D. and Suppes, P. (1965). Preference, utility and subjective probability, in R.D. Luce, R.R. Bush and E. Galanter (eds), *Handbook of Mathematical Psychology*, Vol. 3, Wiley and Sons, New York, NY, pp. 249-410.
- McFadden, D. (1974). Conditional Logit Analysis of Qualitative Choice Behavior. In P. Zarembka (ed) *Frontiers in Econometrics*, New York, Academic Press, pp. 105-142.
- Mitchell, T. (2005). Generative and discriminative classifiers: Naïve Bayes and Logistic Regression. Online draft.
- Pruitt, R. (2009). A Pilot Test of RELR at Premier Bankcard. Presentation given at the 2009 South Dakota SAS Users Conference in Sioux Falls, SD in April, 2009.
- Rice, D.M. (2008), Generalized Reduced Error Logistic Regression Machine, *Section on Statistical Computing - JSM Proceedings 2008*, pp. 3855-3862.
- Rice, D.M. (2009), Reduced Error Logistic Regression, Invited Presentation at Classification Society's Annual North American Conference in June 2009. Can be downloaded from Papers and Presentations page at www.riceanalytics.com.
- Spall, J.C. (2003), *Introduction to Stochastic Search and Optimization*. Wiley:Interscience (1st edition).
- Stokes, M. E., Davis, C. S., and Koch, G. G. (2000), *Categorical Data Analysis Using the SAS System*, Second Edition, Cary, NC: SAS Institute Inc.

APPENDIX 1: SYSTEM REQUIREMENTS – MyRELR for SAS 32 or 64 Bit Windows

This appendix provides requirements for installing and running RELR for SAS for Windows. In general, the requirements are the same as for SAS 9.2 for 32 bit or 64 bit Windows. You must update your system to meet the minimum requirements before running RELR for SAS. We also advise users to read about how their anti-virus software may interfere with SAS on the SAS customer website and perform any remedies suggested by SAS.

Purchase of SAS Base and SAS/STAT 9.2 or Enterprise Guide 4.2 license or later from SAS Institute is required to run MyRELR.

Software Requirements – Microsoft Windows 32 bit or 64 bit operating systems such as:

Microsoft Windows Server 2003 Family

- Microsoft® Windows® Server 2003, Standard Edition
- Microsoft® Windows® Server 2003, Enterprise Edition
- Microsoft® Windows® Server 2003, Datacenter Edition

Microsoft Windows XP

Microsoft Windows 7

Hardware Requirements –

Machines Supported

Intel or Intel-compatible Pentium II class processor (minimum required)

Note that these are the same requirements that SAS suggests for SAS 9.2 in their online documentation. Also, note that SAS® and Enterprise Guide are registered trademarks of SAS Institute and MyRELR™ and ParsedRELR™ are trademarks of Rice Analytics. Aspects of MyRELR are patented in US Patent 8,032,473.

APPENDIX 2: MyRELX INSTALLATION FOR ENTERPRISE GUIDE

The only difference with the installation of MyRELX for Enterprise Guide is that the relrfors library assignment will depend upon how Enterprise Guide 4.2 is configured. If Enterprise Guide is configured for a local SAS server on your machine, then relrfors should be assigned as a project library through the Tools-Assign Project Library graphical user interface where it points to the directory where the sasmacr.sas7bcat and license.sas7bdat files exist (see Steps 1 and 2 in Section 2). The RELX calling program (e.g. relrcall.sas) can be opened in the Open Program menu. Once the RELX calling program and the Assign Project Library icons both appear in the Process Flow diagram, one can link the Assign Project Library to the RELX calling program (relrcall.sas) by right clicking on the Assign Project Library icon. Once this link is established, one can right click on the RELX calling program icon and run RELX. If the Enterprise Guide configuration does not allow a local SAS server on your machine, then the relrfors library will need to be configured on the server where it can run. The SAS administrator will, in this case, need to assign the relrfors library as a new library through the Tools-SAS Enterprise Guide Explorer graphical user interface, where again it should point to the directory where the sasmacr.sas7bcat and license.sas7bdat files exist (see Steps 1 and 2 in Section 2). Once this library is assigned, the RELX calling program can be opened and run and it will find the relrfors library on the server where SAS resides.

APPENDIX 3: BEST PRACTICES/COMMON PITFALLS IN RELR MODELING AND RUNNING MyREL

The following is a list best practices and/or common pitfalls in RELR modeling and running MyREL software.

- 1)The best practice is to use SAS 9.2/EG 4.2 or later. MyREL runs in SAS 9.1.3/EG 4.1, but there was a bug internal to SAS 9.1.3 that would cause problems at very large sample sizes in the optimization run. This error was clearly reported when it happened and it happened at extremely large sample sizes which were beyond where RELR asymptotes in accuracy, but now that SAS has changed their optimization in 9.2 to avoid these problems, one does not worry about these problems in SAS 9.2.
- 2)The best practice is not to pre-impute prior to running MyREL, but to let MyREL do the imputation. Users who pre-imputed with their own imputation reported that MyREL had overfitting and other problems. This is not surprising because RELR's error model is corrupted by pre-imputing. The only way for RELR to get a reasonable error model is for it to calculate t values prior to imputation; this is not possible when users pre-impute. The only possible exception would be if the pre-imputation were actually almost 100% accurate, but this is almost never the case unless one has extremely good business rules for the imputation.
- 3)Users should completely avoid variable names that use the underscore character '_' or 'p2', 'p3' or 'p4'. These characters are used by MyREL to indicate interactions or nonlinear effects. The effect of having input variable names with these characters may be unpredictable.
- 4)RELR assumes that the sample used to construct a model is a representative sample from the population. RELR does allow one to oversample a rare binary condition provided that this is a representative sample; it then performs intercept correction to get a scored model. However, RELR does not perform any weighting such as is done in survey research because weighting may not lead to reliable error modeling. Thus, the best practice is for users to build models based upon population samples that are reasonably representative.
- 5)The ROC AUC is now known to be a very unreliable measure of classification performance and should be avoided. To understand the many problems with the AUC that have been reported by experts including Professor Hand – the foremost AUC researcher, see a brief review of AUC problems by Dan Rice that appeared in KDnuggets in 2010 at <http://www.riceanalytics.com/wsn/page15.html>. We offer the KS statistic and the Classification % Error at the KS statistic threshold as much more reliable measures than the AUC. Additionally, the Brier Score (Avg. Squared Error) is a very well established measure of probability accuracy that we also recommend.
- 6)Some users may find it helpful to visualize interactions effects. This may be accomplished with standard graphics software such as SAS Graph. See the training webinar on Running MyREL at riceanalytics.com.
- 7)One should undersample the majority condition and thus effectively oversample the rare binary condition to avoid bias in variance estimates in the t statistic to give balanced models if one wishes to interpret Parsed models; this is not necessary for Fullbest models but will help processing speed. This should be done using the oversamp abilities within the macro.

Interception correction is applied to get correct probability estimates in the scoring step. See the training webinar on Running and Using MyRELR at riceanalytics.com or see a more detailed discussion about why this a good idea in Appendix 4.

APPENDIX 4: ADVANTAGES OF SAMPLE BALANCING

One of RELR's biggest advantages is that extremely large samples are not necessary for accurate models. Thus, down sampling to achieve greater processing or cost efficiency is an important consideration. Assuming that one can obtain a representative sample of the rare and majority groups in a binary model, it is always much more efficient and accurate to base any down sampling on balancing the size of the rare and majority groups. The single best reason for balancing is that the t statistic used in the RELR error model will not be arbitrary with appropriate balancing, but this will only be realized as a more valid and accurate model if a large enough total sample exists for representative samples of both rare and majority binary conditions, so there still may be many times when such balancing is not possible. This appendix explains why sample balancing should be done whenever possible.

The type of t statistic that RELR employs in binary logistic regression depends upon whether the model is a Fullbest or Parsed model. For Fullbest RELR, that statistic is the same as is used for ordinal and interval category models with more than two levels which is the t statistic that measures the extent to which the correlation between two variables is different from zero. This statistic gives approximately the same result as the classic two independent sample t value in binary models. For Parsed models, RELR employs the Welch t value, as it is obvious that the Welch statistic could be employed. This detail was omitted in the Rice (2008) technical paper because it is actually irrelevant to the calculation of the best Parsed RELR model in a completely balanced sample which does not depend upon the nature of the t value used. With completely balanced models, the Welch statistic will give the exact same result as the two independent sample t statistic and that is what we recommend for Parsed RELR variable selection. The reason is that with balanced samples, the variable selection will be independent of the assumption of equal or unequal variance that differentiates these t statistics. Thus, it will also be independent of any sampling bias which may cause inaccurate variance estimation in the case of the Welch statistic. Fullbest RELR is much less susceptible to these biases, but whenever possible it may be beneficial for processing speed alone to balance samples in Fullbest RELR.

These two t statistic computations will also give approximately the same error model and results with groups that are only slightly imbalanced. However, the classic two independent sample t statistic is susceptible to bias which overstates or understates the probability of false mean differences (more Type I error) due to unequal variances between the two comparison groups with unbalanced samples, so the Welch computation is generally seen as a better estimate with normal distributions that are unbalanced and have unequal variance because the Welch statistic will have far less Type I error bias. On the other hand, the classic two independent sample t statistic is less susceptible to deviations from normality, so it still can actually give less Type I error bias especially when outliers may be present and especially in smaller samples. Additionally, this classic two independent sample statistic is more powerful than the Welch statistic (less Type II error) especially with unbalanced samples, as it can detect smaller mean differences between groups than the Welch statistic although this advantage is diminished with larger samples. Simulation results produced by MiniTab® and available online (Minitab Assistant Whitepaper, 2010) document these Type I vs. Type II error bias tradeoffs between classic two independent sample and Welch t statistics. While these Type I and II error biases are not an important consideration in classic hypothesis testing involving mean differences as long as large samples are employed, even small subtle error biases across many highly correlated variables could possibly affect Parsed RELR variable selection, so these Type I and Type II error biases needs to be considered in RELR's variable selection.

The Welch and two independent samples t statistic give exactly the same t value with perfectly balanced samples. In variable selection, we are interested in obtaining reliable, stable results that do not have inherent biases that would favor either the selection or omission of certain variables related to the type of t statistic used and the degree to which variances are equal or unequal and distributions or normal or not normal. This unbiased scenario where the effective RELR error is not determined by a Welch vs. an independent sample t statistic will happen with perfectly balanced samples. It is obvious that each independent variable could be balanced separately based upon the size of the non-missing rare group, but this is not necessary. The reason is that in this balanced sample situation, any missing values in independent variables would be expected to be equally probable in one dependent variable binary group vs. the other with many variables when the error model has significant impact and thus this bias also would be controlled through the assumption of an equal probability of positive and negative error. Thus, approximately equivalent error modeling results would be achieved simply by balancing based upon the dependent variable binary groups alone. For these reasons, it is preferable to use balanced samples in running Parsed RELR where the majority dependent variable binary group is down sampled relative to the rare group and intercept correction is applied in scoring the model. The oversampling feature is provided within MyRELR for this purpose of balancing the samples, although “oversampling” the rare group is actually a misnomer here as MyRELR is actually down sampling or under sampling the majority group to achieve this balance. With such balanced samples, one largely avoids any Type I or Type II t statistic error bias across variables. The avoidance of Type I or Type II error bias across variables is much more important in Parsed RELR because the variable selection can be affected by even slight Type I or Type II error bias in one variable compared to another with many highly correlated variables. The avoidance of error bias by balancing the sample is not an issue in Fullbest RELR where the models are not interpretable anyway and the large number of variables in a solution would be expected to have the effect of averaging out any bias across variables and would also be less susceptible to missing values, so these bias problems would be effectively muted in Fullbest RELR. While bias problems related to outliers in non-normal distributions and power deficiencies would be less of an issue with very large unbalanced Parsed RELR samples where the Welch statistic shows asymptotic convergence to normal distribution population results (assuming unbiased samples), balancing in extremely large Parsed RELR samples still has the benefit of speeding processing in highly unbalanced samples. This same benefit would be apparent for Fullbest models.

With samples that are not perfectly balanced, the fact that Parsed RELR is based upon the Welch statistic that is less sensitive to Type I error bias with unequal variance in unbalanced models will help it to avoid spurious variable selection better as long as underlying distributions are normal. On the other hand, the Welch statistic’s lower power could theoretically cause Parsed RELR to miss the selection of important variables. Consistent with these possibilities, research on this implementation in Rice (2008) suggests that Fullbest RELR can be substantially more accurate in extremely small samples, whereas Parsed RELR and its error modeling based upon the Welch statistic will likely approach the accuracy of Fullbest RELR in larger samples as more variables are detected in its variable selection. The Rice (2008) samples were almost, but not quite perfectly, balanced and there were substantial missing data in many variables. In that case, it still took about 1100 total observations for Parsed RELR variable selection to pick up enough variables and match Fullbest RELR in accuracy. With greater imbalance in samples, it would likely require a larger sample for this to happen when ample missing data are also present. Another related issue is simply the amount of error in a model. Models with relatively more error such as the Stockdata sample model would be expected to require larger samples for accurate and stable variable selection, whether or not they are built from balanced samples due to the difficulty in detecting a real effect because of the greater error. But, the Rice (2008) models had relatively low error and still required 1100 observations before Parsed approximately matched Fullbest in accuracy.

Balancing the samples will give much more unbiased variable selection than any other approach to down sampling and this would be especially true with ample missing data and/or relatively high error. Balancing will have the effect of negating the

bias issues related to Type I and Type II error that may render Parsed variable selection less accurate and less reliable in down sampled unbalanced samples. It will also make models run much faster, as much smaller total samples often will be possible compared to any other approach to down sampling that maintains unbalanced samples. For these reasons related to speed of processing, balancing may also be beneficial to Fullbest RELR, even though the bias issues are not really an issue there. The only time that balancing may not be preferable is when there is not a large enough sample for it to give reliable down sampling in which case the total sample would be preferred in Fullbest RELR and that total sample would not be large enough anyway for very reliable and accurate Parsed RELR selection. However, assuming that the total sample is large enough and is completely unbiased across all variables, the unbalanced Parsed RELR error model and classification accuracy would be approximately the same anyway. This is because the error model is determined by the relative t values across variables rather than the absolute values and these would be approximately equal in balanced vs. unbalanced models when representative samples are employed. Thus, a balanced sample model is simply a much more efficient approach to getting an optimal model assuming that the down sampling is representative, as total unbalanced samples vs. down sampled balanced samples with intercept correction at scoring will yield approximately equivalent solutions.

Even though the Welch statistic would converge to the asymptotically normal and correct estimate with a very large and unbiased or representative sample, the reality is that a completely unbiased sample will be very difficult to attain. For example, there are certain variables such as those involving repeated measurements that may be expected to have equal variance in the two binary groups in a binary RELR model because each binary group's data sample is from the same population source. Yet, it is very likely that there could be a bias in Welch statistic's estimate of the variance in each group even with a very large sample, so equal variance will not be observed. For example, a cyclical variable such as related to the business or economic cycle like a country's unemployment rate or a store's sales or a person's job security, may be sampled too much at one phase vs. another in the business cycle and there may be no way to know if this measurement bias has occurred. Thus, any attempt to measure the variance objectively even at an extremely large sample size will fail and the Welch t estimate will also be in error. By balancing the samples in a binary RELR model, one knows that any misestimate of variance will have no effect on the solution for any variable whether there is actually equal or unequal variance in the two binary groups. The Welch statistic was employed for Parsed RELR because it is better at avoiding spurious variable selection in unbalanced samples. Spurious variable selection is a much greater annoyance than under fitting and missing actual important variables, but such under fitting is the disadvantage of Parsed RELR when it uses the Welch statistic with unbalanced samples. Provided that the sample is large enough, a balanced sample should be employed that would make it irrelevant what t statistic is employed in Parsed RELR and that is why we recommend balanced samples.