ABSTRACT
In regression, we often have several explanatory variables. Sometimes we are asked to cull these variables to arrive at a “parsimonious model” or “best model.” The goal is to maximize explanatory power while minimizing the number of variables included in the model. But exactly which variables should be selected for inclusion? If there are \( p \) variables, then there are \( 2^p \) possible combinations of those variables. Because our time is finite, search heuristics we typically employ evaluate only a small subset of those possible models.

Common model selection methods focus on incrementally adding (or subtracting) variables and assessing their fitness for inclusion. PROC REG has nine model selection methods, including the automated algorithms of forward, backward, and stepwise selection. PROC GLMSELECT adds LAR and LASSO to these methods.

Ideally, explanatory variables each have a relationship with the outcome variable that is linear (with respect to the link function) and independent of the other variables being assessed. In practice, explanatory variables’ relationships with the outcome variable may be confounded by one another. Thus a better strategy to arrive at an optimal model may be to assess variables as sets and not individually.

This paper demonstrates a genetic algorithm (GA) search heuristic using SAS. Sets of variables are assessed for fitness. The sets with the greatest fitness are selected to undergo modification through genetic operations-- such as mutation and crossover-- to create a generation of offspring sets. The selection and modification processes repeat until an optimal or near-optimal model is identified.

WHEN TO SEARCH FOR A PARSIMONIOUS MODEL
REASONS TO SEARCH FOR A PARSIMONIOUS MODEL
Before embarking on a search for a parsimonious model, consider why you want to conduct the search. This will help you decide how to tune your selection parameters. Some examples of why you might want to determine a parsimonious model:

- To describe the relative importance of variables: Among a large set of parameters, suggest which may be the most useful in predicting the outcome variable.
- To save money in data collection and management: Since obtaining and managing data on variables has nonzero cost, suggest for which variables would it be most useful to have data.
- To enhance predictive ability: A model with a large number of predictors may “overfit” the data on which it is built, hurting its ability to predict the outcome variable when presented with novel data.

The decision to search for a parsimonious set of predictors should be considered carefully, particularly against the alternative of fitting a maximum (aka full) model, where all predictors are employed. With sufficient statistical power, a maximum model may yield better (i.e., more valid) parameter estimates, since the model can then control for the effect of additional, although non-significant, variables that may be confounders. Another good alternative may be the use of expert knowledge to determine the set of parameters to be included in a parsimonious model, which may yield a more convincing model.

REASONS NOT TO SEARCH FOR A PARSIMONIOUS MODEL
Some drawbacks to searching for a parsimonious model are that:

- Inference from traditional statistical tests (e.g., F tests) for the model will be compromised due to the multiple comparisons made to discover the model.
- The search may not necessarily yield the absolute optimum within the space of possible candidate models. More parsimonious or more optimal models may remain undiscovered.
- The time and resources required to divine the model may not be worth the effort.
• Using an automated algorithm to arrive at a parsimonious model may yield a model that belies reason—possibly appearing contradictory or erroneous—and may lead to invalid inference.

WHY USE A GENETIC ALGORITHM?

In regression, given \( p \) possible parameters—among them an intercept term—there are \( 2^p \) candidate models. If \( p \) is sufficiently small, then it is possible to evaluate each of the possible models to find an optimal model. But when \( p \) is sufficiently large, the number of possible models from which to select a parsimonious model is too large to evaluate within a reasonable amount of time. In such cases, the usual strategy is to use a well-chosen heuristic to search for the optimal model. Because this endeavor, in the parlance of computer science, is an NP-hard problem, the absolutely optimal model may not necessarily be discovered, but it or a near-optimal model can usually be identified.

Model selection heuristics commonly taught and used in statistical practice—such as forward, backward, and stepwise—incrementally add (or subtract) parameters from the model. Proposed at a time when computational power was far more limited, these heuristics have many flaws:

1) Selection of parameters can be arbitrary. For example, two variables may have almost exactly identical explanatory power, but one is excluded from a parsimonious model because its p-value is 0.052, while the other is included because its p-value is 0.048.

2) The space searched is too small. For example, forward selection will only evaluate \( p(p+1)+1 \) models—in Big O notation, \( O(p^3) \). As \( p \) increases, the proportion of space searched using this heuristic becomes infinitesimally small. Even when \( p \) is not overly large, forward selection will evaluate far less than 1% of the space (e.g., if \( p=20 \), only 0.02% of the space will be searched).

3) The space searched is constrained. Early decisions strongly influence which models are ultimately explored. For example, the first two variables chosen for inclusion may in fact poorly reflect the underlying actual model, but since they are already chosen they may influence the remainder of other variables chosen in an unfortunate way.

In an ideal world—one in which the model assumptions are all met precisely—all well-chosen heuristics will discover the optimal model. But in the real world, model misspecification and variability can prevent the discovery of the optimal model. Confounding, endogeneity, and nonlinearity (with respect to a link function) alone or in tandem can create an absolute optimum sufficiently distant from other local optima such that heuristics that search by incrementally adding or subtracting parameters will fail to discover it.

This reality has led to the use of nonparametric and nonlinear search heuristics that together are often designated as “data mining” techniques. Many analysts, however, prefer to use the traditional regression heuristics found within the family of generalized linear models, because of familiarity with these methods and for the reporting of parameter estimates amenable to comprehensible interpretation.

The challenge is, in a reasonable amount of time, to search a sufficient number of models to find a model that is either close to or is exactly the best model. Having only the most promising candidates presented for evaluation would help with this goal. Iteratively, through mutation and crossover, a genetic algorithm broadens the field of promising candidates and through selection narrows the field in a search for an optimal model. It is agnostic to the desired regression model, and, when tuned appropriately, avoids the drawback of incremental movement common to many search heuristics, including stepwise, least angle regression (LAR), and least absolute shrinkage and selection operator (LASSO).

GENETIC ALGORITHM, EXPLANATION

At the heart of a GA—paralleling natural selection—is the idea that models that demonstrate lack of fitness should fail to pass on their traits to future generations, while models that demonstrate superior fitness should be selected for reproduction, producing offspring. These offspring, which combine the traits of their parents, may be even more fit. In successive generations produced through an iterative process optimal models are approached.

Operationally, models are represented as vectors of bits, where the bits represent genes and the vectors represent chromosomes. For example, if the possible parameters are \( \textit{alfa}, \textit{bravo}, \text{and} \ \textit{charlie} \), and the model to be evaluated included \( \textit{alfa} \) and \( \textit{charlie} \) but not \( \textit{bravo} \), then the bit-vector chromosome for that model could be represented as \( 101 \), where the digits are Boolean variables that indicate, respectively, the presence of \( \textit{alfa} \), the absence of \( \textit{bravo} \), and the presence of \( \textit{charlie} \) in the model.
If a particular parameter proves useful in predicting the outcome variable, then chromosomes containing that parameter are more likely to be selected for reproduction, and thus produce offspring featuring that parameter.

SPECIFYING A FITNESS FUNCTION
A genetic algorithm relies on the thoughtful specification of a fitness function. Values of this function determine how likely a particular chromosome will be selected for reproduction. Properly designed, a fitness function is an objective function that rewards increases in the explanatory power of a model, while penalizing increases in the number of parameters used by the model. For example, in Vinterbo and Ohno-Machado (1999), they suggest that for logistic regression the reward term be the concordance index (c-index) suggested by Harrell, et. al. (1982) and the penalty term be \((p - n)/p\), where \(n\) is the number of parameters in the model and \(p\) is the number of candidate parameters.

The selection of a fitness function is at the analyst’s discretion. Depending on her preferences, she may choose to increase the penalty for the inclusion of additional parameters—apt when data collection is costly—or to lessen the penalty—appropriate when additional explanatory power is desired.

Balancing the weights given to reward and penalty terms can be tricky. Fortunately, information theory offers helpful suggestions as to what may be appropriate. For information criteria like Akaike’s information criterion and the Bayesian (aka the Schwarz) information criterion, the reward term is \((a \text{ linear function of })\) log likelihood and the penalty term is \(n\), the number of parameters in the model. The weights given to the functional forms of the terms for these criteria derive from statistical deliberation.

Using an information criterion as the fitness function perforces means that parameters are evaluated as a set, and not as individual elements. This provides an opportunity to include variables with important influence that nonetheless do not reach a particular threshold for statistical significance and, possibly, in part, address the possibility of confounding or synergy across parameters. Using the paradigm of genetics, the value of the fitness function for a particular model can be considered a phenotype (an observable property) of the model’s genotype (the set of parameters that constitute the model).

GENETIC ALGORITHM, OUTLINE
Implementation of a genetic algorithm in SAS® requires an understanding of the macro language. Particularly useful is the use of the &&VAR&I construct, as described in Tsai (2008).

Below, I will describe an implementation of a genetic algorithm using SAS® in three ways: [1] in text, [2] as a SAS code outline containing comments and a skeleton of macro code, and [3] as a program flow chart as depicted in Figure 1. I suggest that you, while reading [1] the text, refer to [3] Figure 1 and [2] the SAS code. In the text below, boxed headings correspond to algorithms displayed within boxes in the program flow chart, and the bold subheadings under those headings correspond to boxed comments within the SAS code.

GENETIC ALGORITHM, IN TEXT

INITIALIZATION

CREATE &&VAR&I MACRO VARIABLES FROM THE LIST OF POSSIBLE PARAMETERS
From the list of possible parameters, create &&VAR&I variables so that the parameters are indexed. Save also the number of possible parameters as a macro variable for use in %DO-%END looping.

CLEAR WORKSPACE
Delete data sets that store data from prior runs of the algorithm, if so desired.

INITIALIZE PARAMETERS
Assign default values to macro variables if values have not already been assigned.

EVALUATE AND STORE MINIMUM (AKA NULL) MODEL
The minimum model— the regression model with only an intercept— can aid the choosing of selection criteria and help establish a floor for minimum explanatory power.

EVALUATE AND STORE MAXIMUM (AKA FULL) MODEL
The maximum model— the regression model with all parameters, including an intercept— can aid the choosing of selection criteria and help establish a target to surpass.
INITIALIZE PARENTS
The genetic algorithm begins with an initial generation of parent chromosomes. Parents are randomly generated (bit-vector) chromosomes with each bit turned “on” or “off” according to a non-zero, non-certainty probability. Starting with a large number of parents—perhaps much larger than the size of later generations—furnishes the opportunity to explore a great deal of the search space before selection begins to narrow the areas in which an optimum will be sought.

EVALUATION OF INDIVIDUALS’ FITNESS & STORAGE OF INFORMATION

EVALUATE INDIVIDUALS
For each gene, calculate its fitness function value. Create a data set to contain that generation of chromosomes and their fitness function values. Also, create a data set containing unique chromosomes and their fitness function values. As the algorithm proceeds, a chromosome may already have been evaluated. You can save time by looking up its fitness function value rather than calculating it.

EVALUATION OF GENERATION

ASSESS GENERATION
Evaluate and store historical information on the characteristics of the generation, e.g., the mean fitness function value.

REPORTING, INTERIM
Report the generation number, the most optimal model yet discovered, and the characteristics of the generation.

EVALUATE ESCAPE CRITERION
The genetic algorithm loops and thus requires an escape condition to terminate the algorithm. In Vinterbo and Ohno-Machado (1999), they terminated when the mean fitness function value for a particular generation failed to exceed any of those for a specified number of generations immediately preceding. Other reasonable criteria for termination include: failure to surpass the most optimal model after a specified number of generations, reaching time or resource constraints, or surpassing of a minimum fitness level.

SELECTION

SELECT THE FITTEST CANDIDATES
Using the fitness function values for a generation, employ an algorithm to select among them candidates for reproduction. Those with superior values should be given preference in the selection for reproduction.

The method of selection is at the analyst’s discretion. One popular method used for genetic algorithms is stochastic universal sampling (Wikipedia 2009).

REPRODUCTION

The goal of reproduction is to yield offspring that inherit significant explanatory power from their parents, yet of sufficient diversity to explore possibilities beyond those offered by incremental alterations. The genetic operation of crossover attempts to combine the best genes from two parents into their children, while the genetic operation of mutation allows exploration of the model space beyond strict inheritance. The tuning of crossover and mutation probabilities can greatly affect how soon the algorithm meets its escape criterion, so the probabilities should be carefully chosen.

CROSSOVER
Crossover is when two chromosomes swap a set of contiguous genes. For example, if one chromosome is 11110000 and the other is 10101010 and they swap the genes in positions 2 through 5, the resulting offspring will be: 10101000 and 11110110.

MUTATION
Mutation is when any particular gene may be undergo a change in value, i.e., its Boolean value may be changed, viz. a 1 is changed to a 0 or a 0 is changed to a 1. Commonly used values for the gene-specific mutation probability range from 10% to 15%.

PREPARE NEXT GENERATION
Aggregate children (and, optionally, parents) to form the next generation for evaluation.
REPORTING, FINAL

Report summary information, including the parameters of the most optimal model discovered.

CONCLUSION

In the search for an optimal model (usually a “parsimonious” one), the significant advantage of employing a genetic algorithm is that the search is not constrained to incremental modifications to a base model, fostering deeper exploration of the model space. Using a GA allows analysts to use familiar regression models with interpretable parameter coefficients, instead of possibly unfamiliar nonlinear or nonparametric heuristics with inexplicable coefficients. In addition, the GA method is agnostic to the regression model chosen, so can be used in situations in which there is scant guidance in the statistical literature. The natural GA experiments we observe in the natural world suggest that GAs have utility when searching for optima. Accordingly, when searching for an optimal model, as an alternative to more traditional, and often incremental, search heuristics, consider implementing a genetic algorithm.
GENETIC ALGORITHM AS PROGRAM FLOW CHART

Figure 1: Program flow chart

 Initialization

 Evaluation of individuals’ fitness & Storage of information

 Evaluation of generation

 Reporting, interim

 Evaluate escape criterion

 Selection

 Reproduction

 Reporting, final
GENETIC ALGORITHM AS AN OUTLINE OF SAS CODE

Below is the skeleton of an SAS macro that implemented the genetic algorithm used in Fung et. al. (2009).

*---------------------------------------------------------------------*;
* MACRO GENETIC()                                                      *;
*---------------------------------------------------------------------*;
* Genetic algorithm for optimal model discovery                       *;
*---------------------------------------------------------------------*;
* SEED     = Random number seed                                       *;
* INPUTDS  = Analysis data set                                        *;
* PROCCODE = Complete procedure code (e.g., PROC SURVEYREG,           *;
*           PROC LOGISTIC, etc.), representing the list of            *;
*           parameters with a question mark (?) wildcard              *;
* VARLIST  = List of variables to evaluate                            *;
* STARTSZ  = Size of starting population                              *;
* SELECTSZ = Size of desired selection population (i.e., number of    *;
*           survivors that parent the next generation)                *;
* MINMAX   = Whether the (adjusted) fitness function needs to be      *;
*           MINimized or MAXimized                                     *;
* XOMU_SEQ = Flag that decides whether the children generated by     *;
*           crossover of the selection population undergo mutation or  *;
*           whether a separate set of children are generated from     *;
*           the selection population through mutation. Y means the    *;
*           former (i.e., crossover and mutation are in sequence,     *;
*           while N means the latter, i.e., separate sets of         *;
*           crossover children and mutation children are generated.)  *;
* XOVERTYP = Flag as to whether a (F)ixed number of pairs are selected *;
*           for crossover or whether the pairs are selected according  *;
*           to a defined Bernoulli (P)robability                      *;
* XOVERP   = Used only if &XOVERTYP. = P. The probability that a       *;
*           randomly selected pair of vectors is selected for         *;
* XOPAIRS  = Used only if &XOVERTYP. = F. The specified fixed number  *;
*           of pairs among all possible pairs between members of the   *;
*           selection population that are selected for crossover      *;
* MUTATTYP = Flag as to whether a (F)ixed number of bits are chosen   *;
*           for mutation or whether the bits are selected according    *;
*           to a defined Bernoulli (P)robability                       *;
* MUTATP = Used only if &MUTATTYP. = P. The probability that a randomly selected bit on a vector is selected for mutation. *
* MUTPERV = Used only if &MUTATTYP. = F. The specified, fixed number of bits in each vector that are selected for mutation. *
* PARKEEP = Flag for whether to retain the selected parents (i.e., the selection population) as part of the next generation along with their children) (default Y) *
* CRITER = The specified criterion that determines the fitness function used. (levels: CP, AIC, etc.) *
* ADJ_FORM = The literal formula used to create the adjusted fitness function (variable ADJ_FF) *
* SELMTHD = Selection method (levels: SUSAMP = stochastic universal sampling) (default SUSAMP) *
* MINNUM = Minimum number of generations (not including the starting population known as Generation 0) that must be evaluated (default 30) *
* CHKNUM = Number of generations for which no improvement in the mean fitness function of a generation has occurred (over that which has been observed before that signals the algorithm to discontinue) (default 20) *
* TEMPHALT = Number of generations to run before halting for checking *
* CONTINUE = Whether the algorithm is invoked de novo or as a continuation after a forced halt (see &TEMPHALT.). If &CONTINUE. = Y then the invocation is a continuation from a forced halt. Otherwise, it is a completely new run and all parameters and values will be initialized. (default N) *
* MFF_FILE = Path and filename of text file to hold the mean fitness function value of each generation *
* DEL_DS = Flag on whether to delete data sets (to clean them up) or to retain them (for diagnostic value) (default Y) *
* DIAGPRNT = Flag to print certain parameter values and other values to the log for diagnostic value (default N) *
* Creates data sets:
* _VECTORS *
*----------------------------------------------------------------*

%macro genetic
(   seed     = , inputds  = , proccode = , varlist  = ,
    startsz = , selectsz = ,
    minmax  = MIN , xovertyp = Y , xovertyp = Y ,
    minmax  = MIN , xovertyp = Y , xovertyp = Y ,
    parkeep  = Y , criter  = ,
    adj_form = , selmthd = SUSAMP ,
    minnum  = 30 , chknum  = 20 ,
    temphalt = , continue = N ,
    aff_file = , del_ds   = Y ,
    diagprnt = N )
);

%*-----------------------------------------------------------------*;
%* CREATE &VAR&I MACRO VARIABLES FROM A LIST OF PARAMETER VALUES *
%*-----------------------------------------------------------------*;

%*-----------------------------------------------------------------*;
%* CLEAR WORKSPACE - DELETE STORAGE DATA SETS *
%*-----------------------------------------------------------------*;

%*-----------------------------------------------------------------*;
%* INITIALIZE PARAMETERS *
%*-----------------------------------------------------------------*;

%*-----------------------------------------------------------------*;
%* EVALUATE AND STORE MINIMUM (aka NULL) MODEL *
%*-----------------------------------------------------------------*;
%evalchro(*); %storchro(*);

%*-----------------------------------------------------------------*;
%* EVALUATE AND STORE MAXIMUM (aka FULL) MODEL *
%*-----------------------------------------------------------------*;
%evalchro(*); %storchro(*);
%* INITIALIZE PARENTS
%*/
%prep_bvs(*);

%* ITERATE: CREATE NEW GENERATIONS AND SEARCH FOR OPTIMAL MODEL */

%let critmet  =  0;
%let mff_flag =  0;
%let gennum   =  0;

options nonotes;

/* BEGIN LOOP */
%do %until (&critmet. = 1);
  %* EVALUATE INDIVIDUALS */
  %put RANDOM SEED = &seed.;
  %put Generation &gennum.: ;
  %do vecnum = 1 %to &popsize.;
    %put Assessing Vector # &vecnum.: &&vec&vecnum.. ;
    %* DETERMINE IF THE CHROMOSOME HAS ALREADY BEEN EVALUATED -------*
    %prevseen(*);
    %* DO NOT EVALUATE CHROMOSOME IF IT HAS ALREADY BEEN EVALUATED ---*
    %if &newgene. = 1 %then %do;
      %* EVALUATE CHROMOSOME -------------------------------------------*
      %evalchro(*);
      %end;
    %* STORE FITNESS FUNCTION RESULT FOR SELECTED CHROMOSOME-----------*
    %storchro(*);
  %end;
  %* STORE GENERATION *;
  %storegen(*);
  %* ASSESS GENERATION */
  %evalgen(*);
  %if &mff_flag. = 1 %then %do;
    %let critmet = 1;
  %end;
  %if &critmet. ne 1 %then %do;
    /* IF ESCAPE CRITERION NOT MET, SELECT NEXT GENERATION */
    %* SELECT THE FITTEST CANDIDATES */
    %let gennum = %eval(&gennum. + 1));
%put Selecting generation &gennum.;

%* APPLY SELECTION METHOD;
%selctgen(*);

%* REPRODUCE
%*---------------------------------------------------------------*;
%* CROSSOVER
%*---------------------------------------------------------------*;

%xoverf(*);

%*---------------------------------------------------------------*;
%* MUTATION
%*---------------------------------------------------------------*;

%mutatf(*);

%*---------------------------------------------------------------*;
%* PREPARE NEXT GENERATION
%*---------------------------------------------------------------*;

%end;

/* END NEXT GENERATION SELECTION */

%end;
/* END LOOP */

options notes;

%*---------------------------------------------------------------*;
%* REPORT RESULTS
%*---------------------------------------------------------------*;

%finalrep(*);

%end geneic;
REFERENCES


Fung, Constance; Tsai, Jerry; Lulejian, Armine; et. al. (2009), “Computerised clinical reminders use in an integrated healthcare system,” The Journal on Information Technology in Healthcare, 7:3, 144-159.

RECOMMENDED READING
Many helpful papers on search heuristics and on the use of information criteria by search heuristics are available in the proceeding of various SAS® conferences.

SEARCH HEURISTICS
Robert Cohen, Peter Flom, and David Cassell have discussed the LAR and LASSO heuristics available in PROC GLMSELECT for (general) linear models:


INFORMATION CRITERIA FOR USE IN MODEL SELECTION
Dennis Beal has often discussed linear regression:


Ernest Shtatland and colleagues have examined logistic and proportional hazards regression:


And Jesse Canchola and Torsten Neilands contributed a paper on **mixed models**:


**CONTACT INFORMATION**

Your comments and questions are valued and encouraged. Contact the author at:

Jerry Tsai  
Clintuition  
2117 Greenfield Ave.  
Los Angeles, CA 90025  
Work Phone: +1.310.903.9462  
E-mail: jerry.tsai@clintuition.com  
Web: http://clintuition.com/

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.  
Other brand and product names are trademarks of their respective companies.