Statistical classification of multivariate flow cytometry data analyzed by manual gating: Stem, progenitor and epithelial marker expression in non-small cell lung cancer and normal lung

Daniel P. Normolle, Vera S. Donnenberg, Albert D. Donnenberg

Supplemental Material

Contents

1 Overview

1.1 Introduction

The supplemental material provides both detailed explanations and computer code to implement the methods briefly described in the article. We will use the following terminology, which may deviate from the common but somewhat inconsistent usage in the flow cytometry literature. A flow cytometry assay assesses a limited number ($f < 20$) of *features* measured on hundreds of thousands or millions (d) of events (e.g., cells) that are typically extracted from a limited number (n, typically tens) of *specimens*. These features are then combined to define a number (m) of *markers*, each of which is either present or not present on every cell. Proportions of the numbers of events having or not having these markers are then determined in the gating process to establish the values of a number (p) of *variables* on each specimen, all of which are between zero and one (or, percentages between 0 and 100). Typically, while $p > n$, it is not many orders of magnitude greater, as in contemporary genomics assays. Our proximal goal is to select variables that are associated with a specimen *condition* (e.g., tumor versus normal lung). This is the act of *variable selection*. Selected variables are used to *train a classifier* (in machine learning, this is called *supervised learning*), and the metric of the success of the training is the accuracy of the prediction of condition.

How variables are selected depends on our distal goal: find all variables associated with the condition; find a parsimonious set of variables that predicts a condition, for instance, to create a diagnostic, predictive or prognostic marker panel; select variables to identify cellular pathways activated in a given condition. The entire analysis to achieve these distal goals is beyond the scope of this article, but the variety of goals implies there is no universally best variable selection method, or set of selected variables; the answer, and the means used to achieve that answer, depend on the question.

We propose a workflow for selection and prediction that can be applied to variables extracted from listmode data files, by unsupervised classification or by manual methods. We describe options for steps of the workflow where options are available. We include examples of `R` code to implement the procedure (`R` is an open source, scripted statistical language that is available free of charge[1] and is the *lingua franca* of advanced statistical computation). We select one or more candidate procedures for the workflow steps and explain the reasons for our selection. Then, we apply the workflow to the lung cancer data set described in the article.

1.2 Workflow

We propose the workflow in Figure 1. Steps 1-3 are the conventional flow cytometry data acquisition and gating process, ultimately resulting in variables $v_{i1}, \ldots, v_{ij}, \ldots, v_{ip}$, which represent the proportion of cells from specimen i with some attribute that also have some other attribute j. Each of the specimens also either has or does not have the condition, e.g., case$_i = 1$ or 0. The goal of the workflow is to determine a mathematical function, the *discriminator*, whose input is the values of a subset (possibly all) of the p variables and whose output predicts that the specimen is a case or a control: $f : \boldsymbol{v} \rightarrow \{0, 1\}$ (we describe this process in terms of a diagnostic response, but it can be easily adapted to a prognostic, predictive or risk model).

[1] `http://cran.r-project.org/`

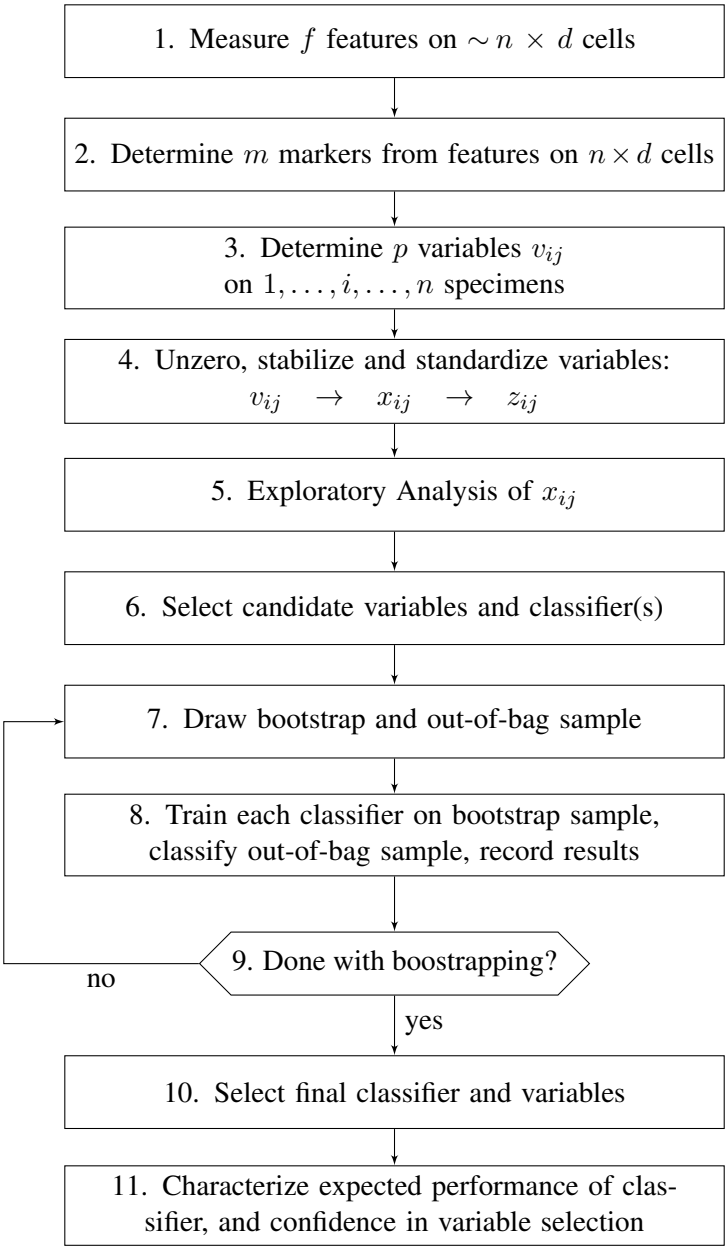

Figure 1: Workflow

"Unzeroing" (Step 4) moves variables of values 0 or 1 ("no cells in this population have this marker" versus "all cells in this population have this marker") slightly away from the boundaries 0 and 1 to facilitate stabilization and to express that a value of zero (or 1) is dependent on the number of cells in the denominator of the proportion; a proportion of $0/100,000$ cells is much more meaningful than a proportion of $0/10$ cells. Stabilization is a mathematical transformation of the variables from the $(0, 1)$ interval to the $(-\infty, \infty)$ real line, eliminating the necessity of constrained optimization and reducing computational issues induced by highly skewed distributions. Standardization is a linear transformation $z_{ij} = (x_{ij} - a_j)/b_j$, where a_j and b_j are chosen so that the mean and standard deviation of $z_{1j} \ldots, z_{nj}$ equal 0 and 1, respectively.

It is useful to assess in an exploratory analysis (Step 5) which variables are highly correlated, and if there is any potential explanatory value in the measured variables. Based on this, some variables may be excluded from the set of the candidate predictors in Step 6, where the candidate classifiers (e.g., linear discriminant analysis, logistic regression, random forest) are also selected. These will be compared in Steps 7-9, an iterative loop in which a *training set* is a *bootstrap sample* drawn from the full sample, the complement of the training set (the "out of bag" sample) is set aside as a test set, all the candidate discriminators are applied to the training set to build discriminators that are then used to classify the test set, and metrics describing the discriminators (sensitivity, specificity, model parameters, tuning parameters, etc.) are recorded. In Step 10, the "best" classifier is selected using one or more metrics, and in Step 11, the ultimate estimates of the Step 10 metrics, and the precision of those estimates, is caclulated.

2 Statistical Methods

2.1 Transforming

2.1.1 Unzeroing

All variables assessed on specimens (v_{ij}), are proportions, with a numerator and a denominator. The meaning of an observed value of 0 (or 1) depends on the size of the denominator. It cannot be assumed that, just because a cell expressing a given marker is not observed in a specimen, it could never be observed, because it could be observable but rare, or the denominator may be so small it is unlikely that even a uncommon but not rare event is likely to be observed. A value of $0/1,000,000$ ($< 1/1,000,000$) indicates an extremely rare event, while a value of $0/100$ ($< 1/100$) only indicates that the true proportion is less than 0.036 (the right-hand side of a 95% exact binomial confidence interval). *Unzeroing* moves $v_{ij} = 0$ (or 1) to the right (left) a small amount to express this uncertainty, and also to facilitate stabilizing transformations of the variables (e.g., the log and logit, both of which are undefined at 0). If there are many zeroes and ones, the analysis may be sensitive to the size of this movement. The choice of unzeroing method is heuristic. We have tried these methods:

1. Replace a zero with a random number drawn from a uniform distribution between zero and one-tenth of the smallest non-zero value; the use of a randomly generated value prevents "piling up" at one value. Values equal to 1 are similarly reduced.

2. Replace a 0 (1) with the right-hand (left-hand) side of a confidence interval. For instance, a 95% exact (Clopper-Pearson) confidence interval for π, given 0 marked cells out of $d = 10^6$ observed cells, is $(0, 3.7 \times 10^{-6})$. More generally, for $d \geq 100$, the upper end of the confidence interval for π is very close to $3.7/d$, suggesting this as a substitute for $p = 0$, where d is the total number of cells in the gate. Similarly, a value of 1 is replaced by $1 - 3.7/d$.

3. Combine the first two methods: replace 0 with a randomly number drawn from a uniform distribution between 0 and $3.7/d$, and replace a 1 with a randomly generated value between $1 - 3.7/d$ and 1.

The first method has the advantage of not depending on d (which may not be available) and not producing spikes in the distribution at $3.7/d$ and $1 - 3.7/d$, while the second and third methods reflect information about the number of cells actually analyzed (if d is available).

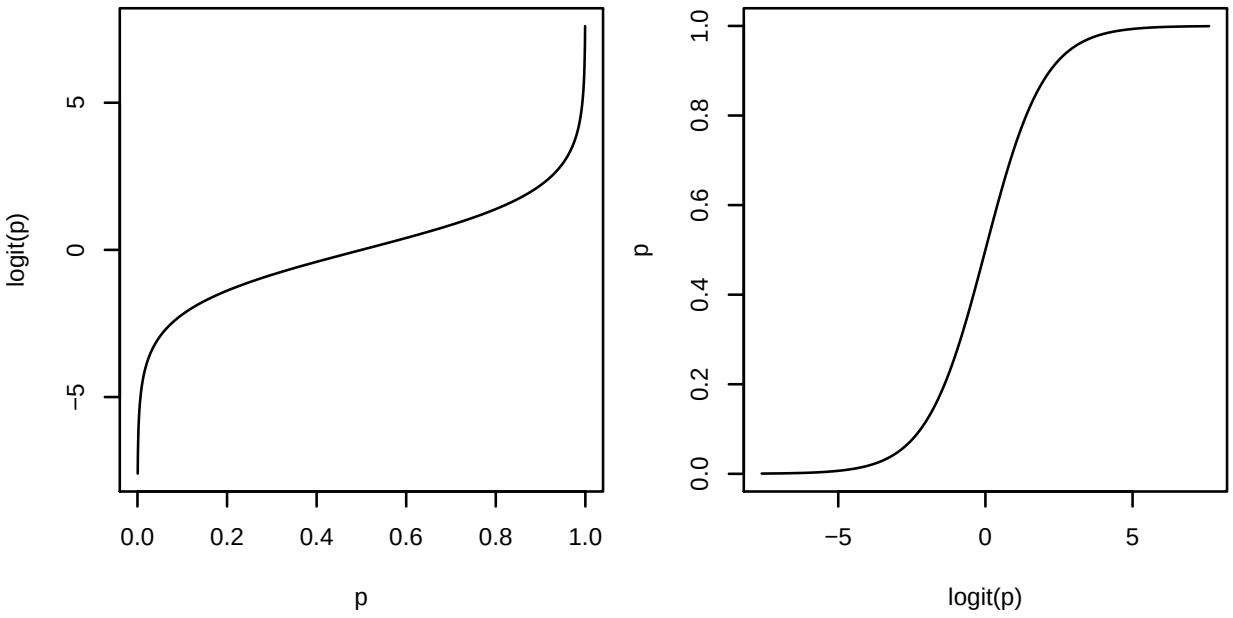

Figure 2: Logit and inverse logit functions

2.1.2 Stabilizing

After unzeroing, the variables v_{ij} are stabilized to make their distribution more symmetric, equalizing the influence of very small (~ 0) and very large (~ 1) values. Transforming a probability with, potentially, a range of several orders of magnitude to a variable with an effective range of $(-14, 14)$ (logit$(10^{-6}) = -13.8$) reduces the leverage associated with very small and very large values. There are several popular stabilizing transformations.

1. While $x_{ij} = \log(v_{ij})$ is well-known, it is asymmetric ($\log(p) \neq \log(1-p)$), mapping the interval $(0, 1)$ to the negative half of the real line.

2. The logit (log-odds) transformation is a natural transformation from probabilities on (0,1) to the entire real line (Figure 2.1.2). The logit of a probability p and its inverse are defined by:

$$x = \text{logit}(p) = \log\left(\frac{p}{1-p}\right) \qquad\qquad p = \text{logit}^{-1}(x) = \frac{e^x}{1+e^x}$$

   The advantage of the logit over the log transformation in this application is that the logit is symmetric (logit$(p) = $ logit$(1-p)$; logit$(1/2) = 0$).

3. The probit transformation ($x_{ij} = \Phi^{-1}(v_{ij})$) is similar to the logit, except for scaling, but the logit is computationally less expensive.

2.1.3 Standardizing

Because they constrain the total magnitude of regression coefficients, some discriminators (e.g., regularized linear regression) require that each variable has a mean of 0 and a standard deviation of 1, but, in any discriminator, the possibility of numerical instability is reduced if all variables are on the same scale. Standardization of each variable x_j is:

$$z_{ij} = (x_{ij} - \bar{x}_j)/s_j \tag{1}$$

where $\bar{x}_j$ and s_j are the mean and standard deviation of all $x_{ij}, \ldots, x_{nj}$, cases and controls.

2.2 Exploratory Statistics

2.2.1 Screening for Outliers

Because the values of the variables are between 0 and 1, outliers in the classic sense are not going to occur, but the unzeroing process, especially Methods 2 and 3, may induce highly influential points. Values can be screened for influence using *robust residuals,* calculated by:

$$e_{ij} = \frac{x_{ij} - \tilde{x}_j}{\text{MAD}_j},$$

where $\tilde{x}_j$ is the median of $\{x_{1j}, \ldots, x_{nj}\}$ and MAD_j equals the Median Absolute Deviation of the j^{th} variable:

$$\text{MAD}_j = \text{median}|x_{ij} - \tilde{x}_j|.$$

As indicated above, the e_{ij} should be calculated on the x_{ij}, prior to standardization. Large positive or negative values of the e_{ij} indicate values that will cause fluctuations in discriminators, especially parametric methods like linear discriminant analysis. What should be done with the values (possibly nothing) depends on what caused them, but specimens with multiple extreme values of e_{ij} should at least be checked for processing errors, such as gating or transcription errors during primary data analysis.

2.2.2 Heat Maps and Dendrograms

In order to visualize the relationships between the variables, we recommend unsupervised clustering on the variables, over all the observations, and over disease-identified subsets (e.g. tumor and normal lung). These can be generated by most heat map software, which will also produce a visualization of strong classifiers or sets of classifiers. Figure 3 displays the heat map created by aggregation clustering of the normal lung and tumor specimens in the lung data set, produced by the `R`[1] function `aspectHeatmap`, which is available in the library `ClassDiscovery`[2], and offers considerable control over the appearance of the graphic. In aggregation clustering, each variable starts as its own cluster, then the two closest clusters are aggregated, and the process is repeated until a single cluster remains. The heights of the cluster joins in the dendrogram indicate the distance between clusters, with the lowest joins being the closest. The heat map provides a quick visual assessment of any strong classifiers of condition, and will be useful when exploring the discriminators promoted by the bootstrap loop, below.

[2] `http://bioinformatics.mdanderson.org/Software/OOMPA/`

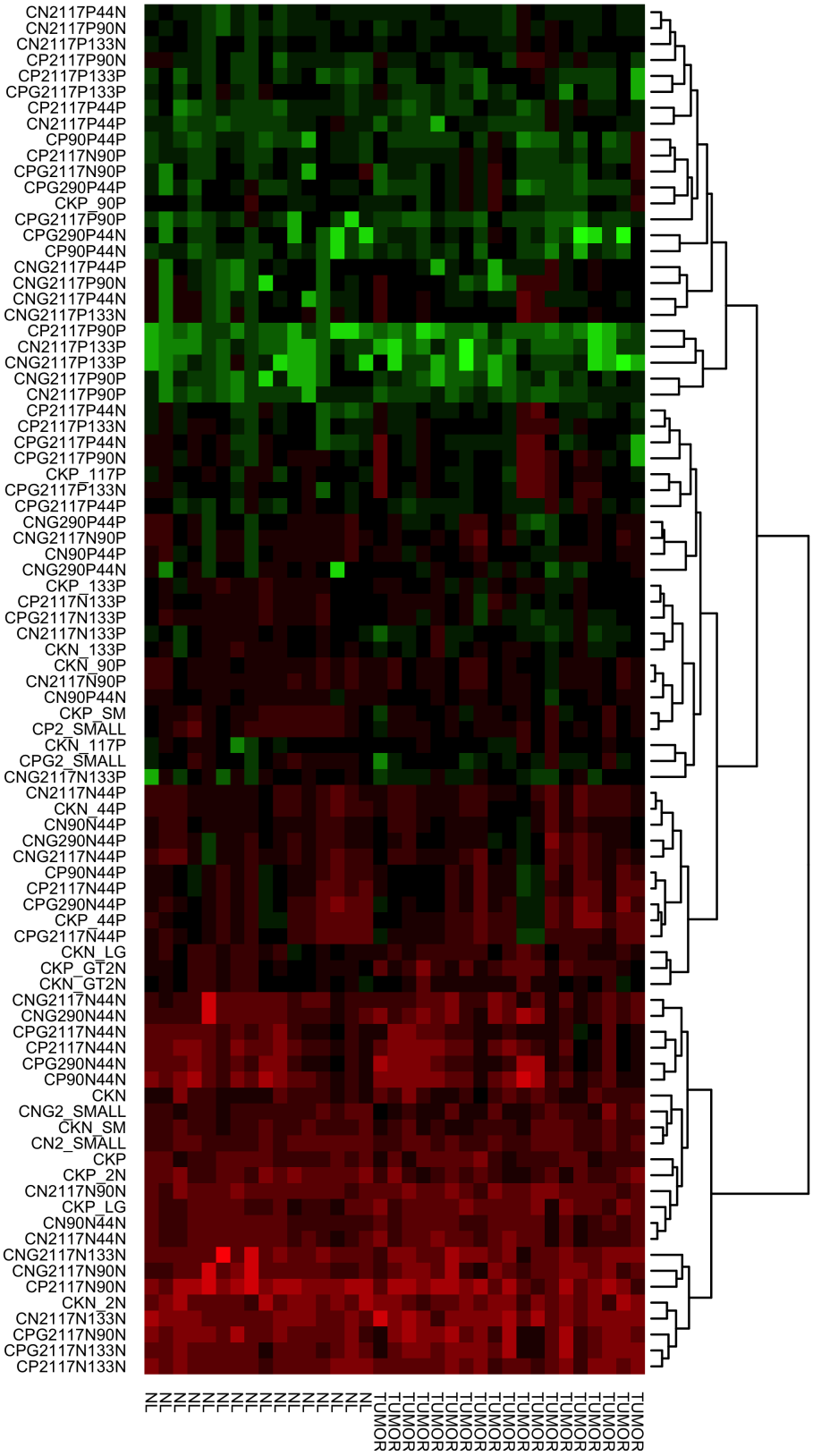

Figure 3: A heatmap of the lung cancer data set. The variable names are defined in Figure 1 and Table 1 of Normolle et al, *Cytometry A* 2012

2.3 Resampling

Resampling methods are key components of parameter tuning, estimation of sensitivity and specificity, and assessment of discriminator stability. All of the classification methods except Single Best Variable require setting a tuning parameter that determines the complexity of the estimated model. In some of the methods described below (e.g., elastic net), tuning by resampling is integral to the process and is implemented automatically; in others (DLDA), we tune using our own `R` code. Even if the sample is large enough to split into a training and validation set, once the validation set has been used, it is no longer independent of the training set, so, when estimating sensitivity and specificity, acknowledging that true independence is not possible, the full set is divided many times into disjoint training and testing subsets, estimating the model parameters only from the training subset and the classification accuracy from the testing subset. After the k discriminations are complete, the sensitivity and specificity estimates can be averaged to achieve a final assessment of how well the condition can be predicted from the variables. Resampling can also be used to choose values of tuning parameters that determine the complexity of the final model; using only the re-substitution estimators of sensitivity and specificity (where all the specimens are used to estimate discriminator parameters, and then the discriminator is used to classify all the specimens) as a quality metric favors more complex models, while estimation based on resampling estimators will moderate that tendency.

Two resampling techniques will be described: cross-validation and bootstrapping.

2.3.1 Cross-Validation

Cross-validated estimates of sensitivity and specificity are nearly as accurate as estimates from holdout, validation sets, while using the entire data set for training. In K-fold cross-validation, the dataset is divided into K exclusive (i.e., non-intersecting), approximately equal-sized subsets, $S_1, \ldots, S_K$, stratified by condition. Typically, K is set equal to a value between 5 and 10, although there is a variant named leave-one-out cross-validation, which effectively sets K equal to n. At each of K cycles, a different subset is set aside to be the test set, and the other $K - 1$ subsets are aggregated to form a training set.

In cross-validation, the set of variables on which the classifier is trained is fixed, and the parameter estimates vary across the cross-validation cycles as the elements of the training set change. At each iteration, the sensitivity and specificity are calculated on the parameter estimates and test set. Once the K cycles are complete, the K sets of classifications are aggregated into estimates of sensitivity and specificity.

Cross-validation is typically used in setting tuning parameters, which do not directly characterize the probabilistic distribution of a population, but provide an empiric description of the ideal complexity of the model. As an example, suppose a tuning parameter λ describes how many of the variables $z_1, \ldots, z_L$ should be included in a discriminator, where the $z_1, \ldots, z_L$ are pre-specified.

The algorithm in Figure 4 uses cross-validation to choose λ, which must be an integer between 1 and L. In this algorithm, as l iterates from 1 to L, the "next best" variable is added to the discriminator based on all the data. After that, the full data set is partitioned into $S_1, \ldots, S_K$, and the discriminator is retrained (i.e., the discriminator parameters are re-estimated) and the accuracy of discrimination is calculated K times $(b_1, \ldots, b_K)$, holding the variable set fixed.

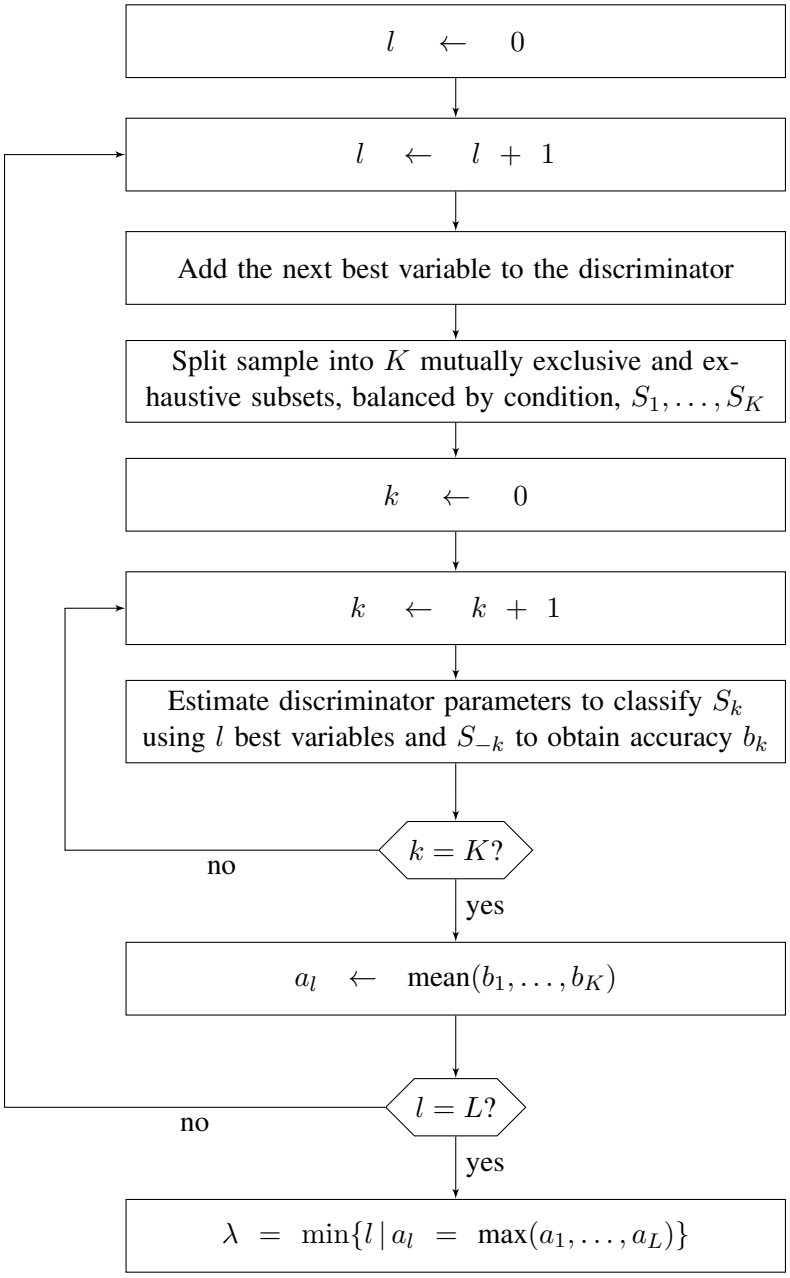

Figure 4: K-fold cross-validation to select the value of a tuning parameter λ. The set S_{-k} is the union of all K subsets of the full sample of specimens except S_k

Then the accuracy is estimated from the mean of $b_1, \ldots, b_{10}, l$ is incremented, the next best variable is identified, the sample is partitioned, and so on. The estimates of a_l will start to increase as l increases, but then will start to decline as more variables are submitted to the discriminator and the training set is over-fit. The recommended value of λ is the smallest l that achieves the best a_l.

2.3.2 Bootstrapping

The key component of of our workflow is one or more statistical classification or machine learning methods, imbedded in a *bootstrap loop* to analyze the performance and stability of the discriminator(s), and to identify influential specimens and important variables. In the nonparametric bootstrap[2], a *bootstrap sample* of observations, the same size as the original sample, is constructed by drawing from the original sample *with replacement*, that is, once an observation has been selected for the bootstrap sample, it is replaced in the pool of eligible observations and may be sampled again. Sampling is stratified by condition, that is, the numbers of cases and controls in the bootstrap sample are identical to the numbers of cases and controls in the full sample. The observations that are *not* selected for the bootstrap sample, approximately $(1 - 1/n)^n \approx 0.37$ of the original sample, referred to as the "out-of-bag" (OOB) observations, are set aside as an independent test sample (they are not resampled). For instance, suppose 16 normal lung specimens are indexed from 1 to 16, and 19 tumor specimens are indexed from 17 to 35. Table 1 shows a single bootstrap sample (left column) and OOB sample. Note that all bootstrap samples will have 16 normal lung specimens and 19 tumor specimens, and some specimens will be included more than once, while the number of OOB specimens in each category will vary, but each specimen will be counted only once.

	Bootstrap	Out-of-Bag
Normal Lung	2, 3, 4, 4, 5, 7, 8, 9, 11, 12, 12, 12, 15, 16, 16, 16	1, 6, 10, 13, 14
Tumor	20, 20, 22, 24, 25, 25, 26, 27, 27, 28, 28, 30, 31, 31, 32, 33, 35, 35, 35	17, 18, 19, 21, 23, 29, 34

Table 1: Example of a bootstrap sample from a full sample where 16 normal lung specimens are indexed from 1 to 6, and 19 tumor specimens are indexed from 17 to 35.

Bootstrapping (Steps 7, 8 & 9 in the Workflow) is repeated B times, where B is usually in the hundreds or thousands: the bootstrap and OOB are drawn; the discriminator(s) trained on the bootstrap sample and the discriminator coefficients are recorded; the OOB samples are submitted to the discriminator(s) and the sensitivity, specificity, and other classification metrics are retained. At the end of the bootstrap iteration, the retained values can be used to analyze sensitivity and caclulate confidence intervals. For example, a 95% confidence interval for a given statistic T can be calculated from the 2.5^{th} and 97.5^{th} percentiles of the B values of T calculated on the B bootstrapped samples.

In the present case, the statistics of interest will include: sensitivity and specificity; the observations which are consistently misclassified; the variables identified as significant. To assess the first and second statistics, at each iteration of the bootstrap loop, the observations *not* selected for the bootstrap sample, approximately $(1 - 1/n)^n \approx 0.37$ of the original sample, referred to as the "out-of-bag" (OOB) observations, are set aside as an independent test sample, the classifier is trained on the bootstrap sample, and then the OOB observations are classified. Estimates of sensitivity and specificity derived by averaging over the B OOB samples are less optimistically biased than using the entire sample to train the classifier, and then classifying the entire set (the *re-substitution estimator*) without adjustment.

The proportion of times over the bootstrap loop that each observation is misclassified is recorded; "problem" observations that are most frequently misclassified can be identified and checked for anomalies. In addition, at each bootstrap sample, the variables selected as important for classification (usually by some test of statistical significance) are recorded to assess the importance of the variables and the stability of the model.

In addition, the sensitivity of the classification to any tuning parameter set by the analyst (such as the mixing parameter in the elastic net, below) can be described by observing changes in the bootstrap estimates of sensitivity, specificity and variable selection probabilities as the value of the parameter is changed.

2.3.3 Variable Selection

The selection of variables from a universe of candidates is a point estimate of a population attribute, just as the sample median and mean estimate the center of a population probability distribution function. Point estimates should be accompanied by an assessment of the precision of the estimate. We address this with a variable importance index available at essentially no computational charge with the bootstrap.

2.4 Classification

If we are interested only in selecting the k variables most strongly associated with a condition, and not trying to characterize the relationship between the variables and the condition, a straightforward analysis plan is available: a nonparametric test (e.g., Wilcoxon's Rank Sum Test) produces a p-value for the association of each variable with the condition of interest, the p-values are ranked from smallest to largest, and the variables associated with the k smallest p-values are selected. It is not necessary to adjust for multiple tests, since the p-values are used only for ranking the variables, and the study specimens are unlikely to have been drawn at random from any specific population (any publication of the k p-values, however, should include, at minimum, the total number of tests performed). But, if the analytic goal is a reduction in dimensionality or a functional model of the condition in terms of a subset of the variables, some sort of variable selection method must be employed with a discriminator. A discriminator is first "trained" on variables derived from specimens of known condition, and will then able to make predictions about the conditions of specimens by their variable values.

There are a wide variety of discrimination (also, "classification" or "supervised learning") methods available, including linear discrimination, diagonal linear discrimination, stepwise linear discrimination, quadratic discrimination, logistic regression, stepwise logistic regression, k-nearest neighbors classification, recursive partitioning, random forest, neural networks, support vector machines and the lasso. A comprehensive review of the features of all these methods is beyond the scope of this article. The classifier(s) selected for evaluation must be chosen considering the following features of flow cytometry data: the data sets are relatively small; the number of variables is larger than the number of observations; the variables are correlated, some highly so; some of the variables, possibly many, are irrelevant to the classification. We will focus on three methods that reflect different philosophies of model building: diagonal linear discrimination (DLDA), random forests and regularized linear regression (lasso and elastic net).

In most of the classifiers, training involves the estimation of a function derived from either assumptions about the probability distribution of the variables, dependent on the condition, or about some feature of the partition itself. An exception is random forest, which partitions the variable space, but, like k nearest neighbors, does so by direct reference to the training set, unmediated by assumptions about probability distributions or the form of the partition. Some of the classifiers have an inherent variable selection method, but, in others, variable selection is a process external to the partitioning. There are three sets of methods for selecting variables:

Filtering followed by classification. A decision rule that can be applied to one variable at a time is used to select the most likely variables, and then a classifier is constructed that uses all the variables so

selected. A common (but not necessarily very good) method is to choose $k < \min(n, p)$ variables as above, and then use Fisher's linear discriminant analysis or logistic regression to predict condition based on the k variables.

Wrapping a classifier in a variable selection scheme. An iterative loop starts with one variable, or all variables, and adds or subtracts variables based on the characteristics of the variables used by the classifier and the variables not used by the classifier. The training is repeated at each iteration. Examples include stepwise linear discriminant analysis and stepwise regression. The loop must include a stopping rule, since knowing when to stop is possibly the most important feature of conducting any activity successfully.

Embedding variable selection in the classifier, which selects variables as part of the estimation of parameters (typically, by setting some variable-specific parameters equal to zero). The use of p-values to select variables in a logistic regression, and then dropping the variables with $p > 0.05$ is *not* an example of an embedded method, because the logistic regression parameters must be re-estimated after the non-significant variables are dropped (this is a filtering method). Random forest, neural net, lasso and elastic net are examples of methods where the variable selection is embedded in the classification.

DLDA is used with variable filtering to produce a final model, while random forest's and regularized logistic regression's variable selections are embedded. We will also use a naïve classifier, best single variable, with a variable filter.

2.4.1 Diagonal Linear Discriminant Analysis (DLDA)

Linear discriminant analysis (LDA) was introduced by R.A. Fisher in 1936[3]; it is the oldest statistical classifier. The classification boundary is $(\bar{\boldsymbol{x}}_1 - \bar{\boldsymbol{x}}_2)\boldsymbol{S}^{-1}(\bar{\boldsymbol{x}}_1 - \bar{\boldsymbol{x}}_2)$, where $\bar{\boldsymbol{x}}_1$ and $\bar{\boldsymbol{x}}_2$ are the sample means and $\boldsymbol{S}$ is the pooled sample variance, a $p \times p$ matrix. Because $\boldsymbol{S}$ will not be invertable if $p \geq n$, LDA, as described by Fisher, cannot be used when the number of variables exceeds the number of samples, and the method has no implicit variable selection mechanism. Diagonal linear discriminant analysis (DLDA) was introduced[**?**] to accommodate data sets where $p \gg n$, by substituting the diagonal matrix $s^2_{\text{DLDA}}\boldsymbol{I}_p$ for an estimate of the full variance matrix $\boldsymbol{S}$ in the LDA discriminator, where s^2_{DLDA} is an estimate of the variance *common to all the variables*; we used:

$$s^2_{\text{DLDA}} = \frac{1}{p}\sum_{j=1}^{p} \frac{n_N s^2_{jN} + n_T s^2_{jT}}{n_N + n_j},$$

where n_T (n_N) is the number of tumor (normal lung) specimens and s^2_{jT} (s^2_{jN}) is the usual estimate of the variance of the values of variable j in the tumor (normal lung) specimens. Because the samples are standardized around a common mean, s_{DLDA} will tend to be a number close, but not exactly equal to, 1. To select variables, we calculated the standardized distance between the normal lung and tumor samples for each variable:

$$d_j = \frac{\bar{z}_{jT} - \bar{z}_{jN}}{s_{\text{DLDA}}/\sqrt{n_{jT} + n_{jN} - 2}}$$

A p-value was calculated for each d_j,

$$p_j = \text{P}(T < -|d_j|)$$

where T is a random variable with a Student's *t* distribution on $n_{jT} + n_{jN} - 2$ degrees of freedom. Variables with a p-value less than some cutpoint α^* are selected as significant. The quality of the classifications is

sensitive to α^*; too large a value will overfit the training data set by including irrelevant variables, while too small a value may miss significant variables. We set α^* as follows:

1. Define a range of candidate cutpoints $\alpha = e^{\{-a_1,\ldots,-a_A\}}$, where the $\{a_1, \ldots, a_A\}$ are evenly spaced. We used $a_1 = -7/4$ and $a_{54} = -60/4$, a range determined by trial and error to be useful.

2. For each α, select the variables having $p_j < \alpha$. When α is small, most or all variables will be eliminated; when α is large, the bar is set high and most variables will be selected. Calculate the cross-validated predictive accuracy of DLDA *on the training set* using the selected variables.

3. Choose α^* equal to the smallest α with the largest cross-validated accuracy.

Once α^* was selected, DLDA was recalculated on the entire training set, and the OOB samples were classified. The estimates of the sensitivity and specificity are the average of those estimates over all the bootstrap iterations.

DLDA is similar to a common practice where the individual variables are analyzed by means of two-sample *t*-tests (or, one-way ANOVAs), and the most statistically significant variables are combined to classify the observations. DLDA stabilizes the p *t*-tests (or ANOVAs) by using a pooled variance estimate, and, in our implementation, constrains overfitting by setting the threshold for statistical significance by cross-validation. DLDA works when $p \gg n$, does not suffer the numerical instability of LDA when p is close to n, but can be shown in some examples to be less efficient than LDA.

2.4.2 Recursive Partitioning Trees and Random Forest

Random Forest[6] evolved from the recursive partitioning tree (RPT)[6]. In RPT, for each variable j, the cutpoint c_j that best separates the cases from controls is determined. Then, the best cutpoint is determined from among the c_j. The sample space is partitioned on this dimension at this cutpoint, producing two p-dimensional subspaces. This procedure is repeated independently in each subspace, to produce four partitions. The four partitions are then split, and the process is repeated in all subsequent partitions until some purity criterion is achieved, resulting in the familiar decision tree (Figure 5).

A partition can no longer be split when it contains all cases or all controls, but partitioning to absolute purity in all subspaces tends to overfit the data. There are a number of heuristics, most of which using some sort of resampling, used to stop the process and produce a parsimonious tree. Recursive partitioning is implemented in the `R` package `rpart` (which was used to produce Figure 5).

The estimates of sensitivity and specificity of RPT calculated by classifying the training set are optimistically biased, and it is easily demonstrated that small perturbations of the training data can produce large changes in the classification tree structure. Breiman addressed these issues by applying the bootstrap to produce many trees (a "random forest"). In random forest, a value $q \ll p$ is chosen. At each bootstrap iteration, n observations are selected with replacement. At each node of the tree, q variables are selected at random, the best split on the q variables is identified, and the partitioning continues until all the nodes are pure (i.e., contain either all cases or all controls). The OOB samples are then classified. This process is repeated many (typically, thousands) of times, and, after bootstrapping, each sample is classified according to the votes from all the trees (over thousands of iterations, all of the samples will be OOB in many trees). Random forest is implemented in the `R` package `randomForest`.

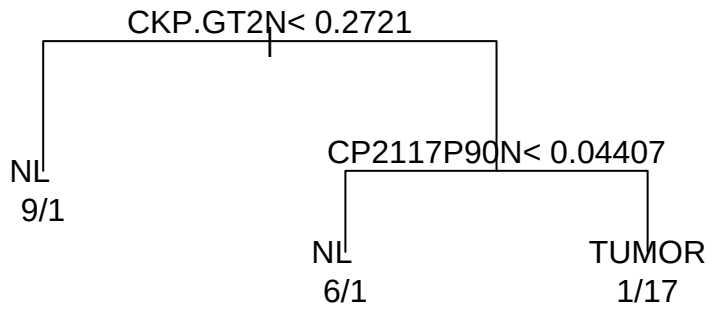

Figure 5: Example of a recursive partitioning tree produced by the `R` function `rpart` in the `rpart` package. If CKP.GT2N<0.2721, classify the specimen as normal lung (NL), otherwise, classify the specimen as normal lung or tumor, depending on the value of CP2117P90N. The quotients indicate the purity of the node; "9/1" indicates 9 normal lung specimens and 1 tumor.

A variable importance index is calculated in the (internal) random forest bootstrap by a permutation test. In each iteration of the bootstrap, after the accuracy of the model is determined by classifying the OOB samples, the values of the first of the q variables in the OOB sample are randomly permuted and the classification repeated. If the first variable contributes significantly to the accuracy of the classification, the degradation of the accuracy by permuting the first variable will also be significant. The difference between the accuracy of the un-permuted and permuted OOB samples is, therefore, a measure of importance of the first variable. This is then repeated for the other q-1 variables in this iteration of the bootstrap loop. After bootstrapping is complete, the average (over the bootstrap iterations) importance is calculated for all p variables, along with a standard error and p-value. We used the variable importance index to select variables by ordering the variables according to their importance index, and then choosing as significant the smallest number of variables associated with the maximum estimated accuracy on the OOB variables.

2.4.3 Best Single Variable

Best single variable discrimination is employed as a reference method. For each variable, the standardized between-mean distance is calculated:

$$d_j = \frac{\bar{z}_{jT} - \bar{z}_{jNL}}{s_j},$$

where s_j is the pooled estimate of the standard deviation for variable j. The variable with the largest distance between means is selected. An observation z_{ij} is classified as normal lung or tumor depending on whether it is closer to $\bar{z}_{jNL}$ or $\bar{z}_{jT}$.

2.4.4 Regularized Logistic Regression

Logistic regression is an example of a *generalized linear model* (GLM), which is an extension of multiple linear regression. Assigning specimens from the two populations the indices 0 and 1 (e.g., $0 =$ Normal Lung

and $1 =$ Tumor), the goal of logistic regression is to estimate the probability that the i^{th} specimen is from a specimen with tumor, i.e.,

$$\pi_i = \mathrm{P}(y_i = 1|\boldsymbol{z}_i)$$

where $\{\boldsymbol{z}_1, \ldots, \boldsymbol{z}_n\}$ are the multivariate standardized specimen-level variables derived from the flow cytometry assay. Logistic regression directly models the log-odds (from which the probabilities can be derived) of a specimen having a tumor as a linear function of the variables

$$\log\left(\frac{\pi_i}{1-\pi_i}\right) = \beta_0 + \sum_{j=1}^{p} \beta_j z_{ij}$$

by maximizing the log-likelihood:

$$l(\boldsymbol{\beta}|\boldsymbol{y}, \boldsymbol{Z}) = \sum_{i=1}^{n} y_i \boldsymbol{z}_i^T \boldsymbol{\beta} - \log(1 + e^{\boldsymbol{z}_i^T \boldsymbol{\beta}}).$$

The parameters $\boldsymbol{\beta} = \{\beta_0, \ldots, \beta_p\}$ are estimated using iteratively re-weighted least squares. As with LDA, and linear models in general, the parameter estimates become unstable as $p \to n$, and, if $p \geq n$, may be non-estimable. *Ridge regression* addresses this instability by adding a constraint to the maximum likelihood estimator:

$$\max_{\boldsymbol{\beta}} l(\boldsymbol{\beta}|\boldsymbol{y}, \boldsymbol{Z}), \text{ subject to } \sum_{j=1}^{p} \beta_j^2 \leq s. \tag{2}$$

All of the β_j in a ridge regression will be non-zero, but those corresponding to variables unrelated to the condition will be very small. If a subset of variables are highly correlated (because, for instance, they represent a set of nodes on a given pathway), their regression coefficients will be of similar size (although possibly of different signs). An alternative is the *lasso*[7], which uses a different constraint:

$$\max_{\boldsymbol{\beta}} l(\boldsymbol{\beta}|\boldsymbol{y}, \boldsymbol{Z}), \text{ subject to } \sum_{j=1}^{p} |\beta_j| \leq s. \tag{3}$$

Lasso coefficients of variables unrelated to the condition will be set exactly to zero, yielding a embedded variable selector. If a subset of variables are highly correlated, one of their regression parameters will be non-zero, and the rest will be set exactly to zero, so, if a set of variables represents nodes on an activated pathway, lasso will pick one of them for the pathway representative, and drop the others.

Usually, these constrained MLEs are expressed in terms of least squares estimators, for ridge regression:

$$\min_{\boldsymbol{\beta}} \sum_{i=1}^{n} [(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j z_{ij})^2] + \lambda \sum_{j=1}^{p} \beta_j^2, \tag{4}$$

and for the lasso:

$$\min_{\boldsymbol{\beta}} \sum_{i=1}^{n} [(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j z_{ij})^2] + \lambda \sum_{j=1}^{p} |\beta_j|. \tag{5}$$

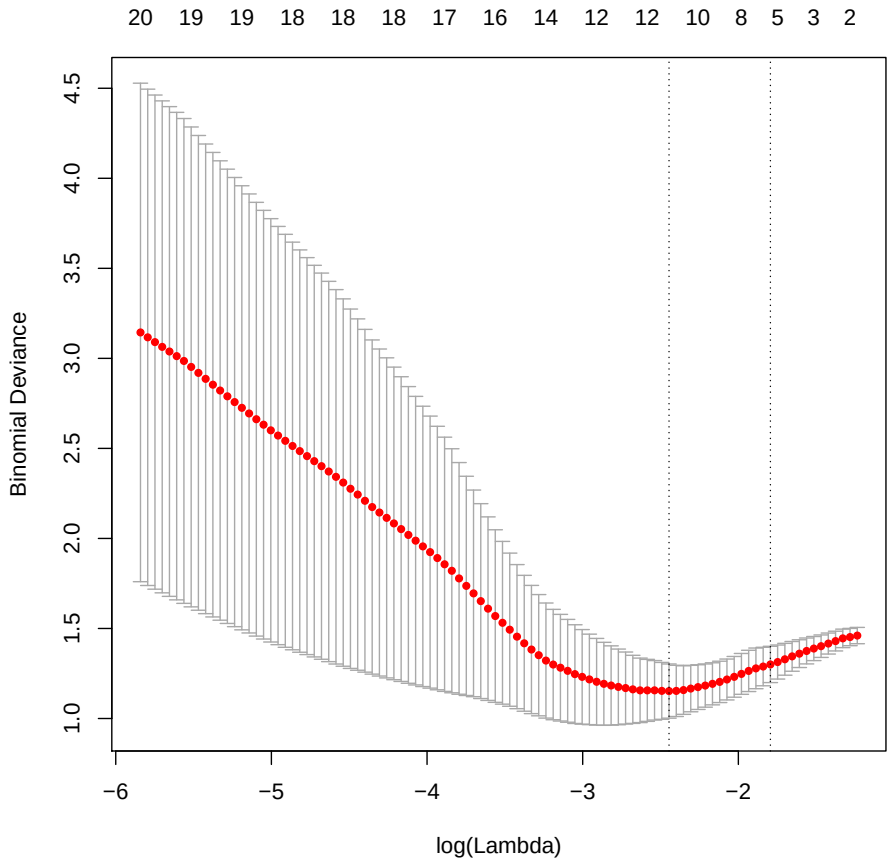

Figure 6: Example results from the R function that optimizes the tuning parameter. The vertical bars are 95% confidence intervals, left dashed line indicates the value of $\log(\lambda)$ associated with the smallest binomial deviance and the right dashed line the largest value of $\log(\lambda)$ such that the binomial deviance is no more than one standard error greater than the minimum.

There is a one-to-one correspondence between s (Equations 2 and 3) and λ (Equations 4 and 5). Usually, λ is set by cross-validation. These two models are subsumed by *elastic net*:

$$\min_{\boldsymbol{\beta}} \sum_{i=1}^{n}[(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j z_{ij})^2] + \lambda(\alpha \sum_{j=1}^{p} |\beta_j| + \frac{1-\alpha}{2} \sum_{j=1}^{p} \beta_j^2)$$

λ is still the penalty parameter, but a second parameter, α, which controls the sparsity of the solution, has been added. It is sometimes called the "mixing parameter," because, if $\alpha = 0$, elastic net is equivalent to *ridge regression,* which will choose all of the regressors, but assign some very small β coefficients, while, as $\alpha \to 1$, elastic net becomes equivalent to the lasso, and will select one regressor from a correlated subset of variables, and ignore the rest. Depending on the value of α, elastic net will choose more of a subset of highly correlated regressors, and assign them similar β values (which makes sense only because the columns of $\boldsymbol{Z}$ have been standardized). The results will still tend to be numerically stable, even if $\alpha \approx 1$. In practice, α is set be the analyst, depending on whether the goal is identification of all variables associated with a condition ($\alpha \sim 0$), or of a parsimonious set of variables (Hastie and Zou[8] recommend $\alpha = 1 - \epsilon$, where ϵ

is a small number, for sparse modeling with numerical stability when $p \gg n$), and, given α, λ is determined by cross-validation. Elastic net[8] implemented in the `R` package `glmnet`[9], which is used here.

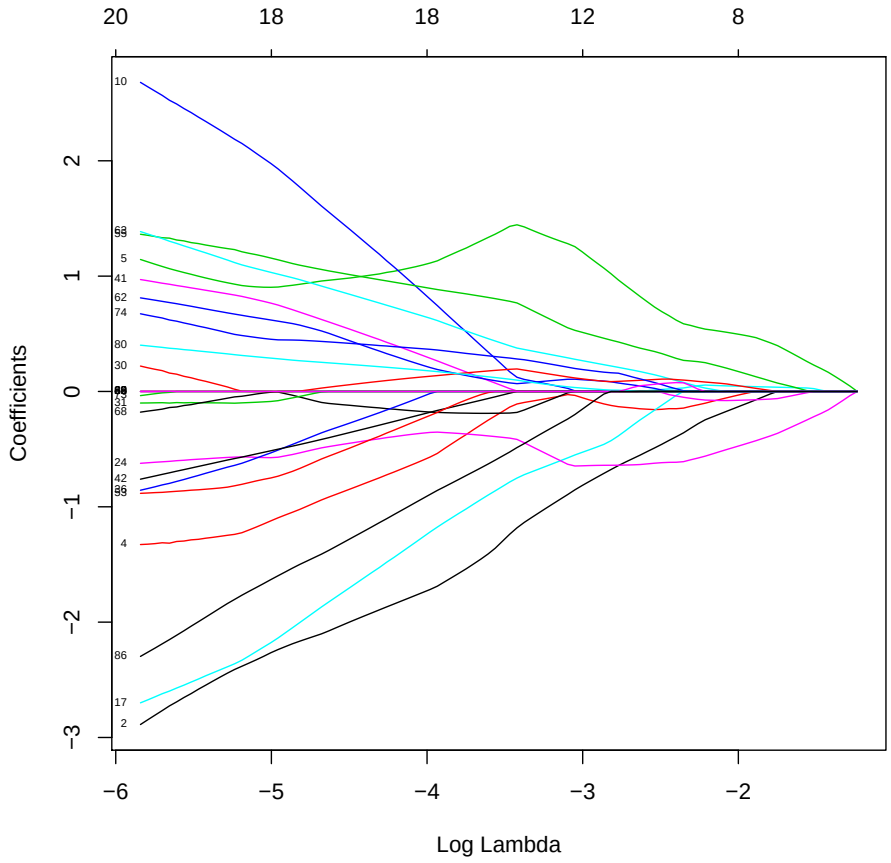

Figure 7: Fit of an elastic net model. The colored lines indicate the sizes of the standardized regression coefficients of the different variables (numbered on the left). A regression coefficient equal to zero indicates the variable is not in the model. Reading from right to left, as $\log(\lambda)$ decreases, more variables enter into the model. At the estimated optimal value of $\log(\lambda)$ (-2.5), ten variables are in the model (numbers across the top).

Parameter Tuning Figure 6 demonstrates the output from the cross-validation process setting λ. The $y-$axis, "Binomial Deviance," is a measure of goodness-of-fit used in logistic regression models; models with better fit have lower deviance. The $x-$axis shows the log-transformed penalty parameter λ, while the numbers across the top indicate the mean (averaged over the cross-validations) number of variables with non-zero coefficients. It is seen, moving from right to left, when λ is large, too few variables are selected, and the model fits neither the training nor testing sets well (in cross-validation, 1/10 of the observations are held out as a test set, and the validation moves through all 1/10 subsets), but, as λ becomes small, the training model is overfit, and the testing set (on which the deviance is calculated) is badly fit. The optimal value of λ is seen to be about 0.08 ($\log_e(0.08) = -2.5$).

2.4.5 Model Interpretation

The results of the elastic net model are depicted in Figure 7. The graph is easiest read from right to left. At the extreme right, λ is so large that the only parameter in the model is the intercept, the estimates of the parameters $\beta_1, \ldots, \beta_p \equiv 0$ and for each sample, the predicted probability that it is a tumor equals the proportion of tumors in the data set, 0.54=19/35. Then, as λ decreases, individual logistic regression coefficients tend to come "unstuck" from 0. Each colored line shows the value of the regression coefficient β_j as λ is decreased. If the line is not visible, $\beta_j = 0$, and the variable is not selected. It is unusual for variables to be deselected, but the change in any given β is not necessarily monotonic as λ decreases. At $\lambda = 0.0025$ ($\log_e(\lambda) = -6$), 20 variables are selected, and the predictions are perfect because the training set is grossly overfit.

2.5 Selected Variables

A variable importance index is calculated in the (internal) random forest bootstrap by a permutation test. In each iteration of the bootstrap, after the accuracy of the model is determined by classifying the OOB samples, the values of the first of the q variables in the OOB sample are randomly permuted and the classification repeated. If the first variable contributes significantly to the accuracy of the classification, the degradation of the accuracy by permuting the first variable will also be significant. The difference between the accuracy of classification of the un-permuted and permuted OOB samples is, therefore, a measure of importance of the first variable. This is then repeated for the other q-1 variables in this iteration of the bootstrap loop. After bootstrapping is complete, the average (over the bootstrap iterations) importance is calculated for all p variables, along with a standard error and p-value. We used the variable importance index to select variables by ordering the variables according to their importance index, and then choosing as significant the smallest number of variables associated with the maximum estimated accuracy on the OOB variables.

3 Results

3.1 Sensitivity and specificity

The sensitivity, specificity and accuracy of the discriminators, estimated over the 500 bootstrap samples, are presented in Table 1. DLDA, random forests, elastic net and the lasso produce essentially equivalent accuracy; random forest is different from the other three in that its estimated specificity is larger than its sensitivity. The differences between sensitivity and specificity in these methods are more likely due to peculiarities of the example set than the methods themselves. Best-single discrimination is somewhat worse than the other methods. Table 1.5 shows the estimates of the same operating characteristics achieved by training the classifier with the whole data set, and then using the same data set as a test set. The bootstrapped estimates of sensitivity, specificity and accuracy are much less optimistic than the resubstitution estimates (training the discriminator once and then using it to classify the entire training set); resubstitution estimates of the accuracy of elastic net, random forest, DLDA and lasso were close to or above 90%.

3.2 Variable selection

Table 4 presents the importance index of each variable on a scale of 0-500 (the number of bootstrap iterations). The values for elastic net, random forest, DLDA and lasso are much larger than those for best-single

discrimination because the total number of variables chosen over 500 bootstrap iterations of best-single discrimination is fixed at 500. There is a fairly large drop in importance after the eleventh variable, CKN_GT2N. The five most important variables do not cluster together (Figure 3), suggesting that these variables represent different processes. Markers commonly chosen are mostly coherent on the average, but different methods can be fairly discordant on the same bootstrap sample. Table 5 int he primary article displays the concordance of the five methods in choosing CKP_GT2N across the bootstrap samples. Even though elastic net and lasso are similar in concept, they are discordant in 86/500=17.2% of the bootstrap samples. Best-1, which is not smoothed or regularized in any way, is usually discordant with the other methods.

3.3 Number of variables selected

Figure 4 displays the distributions of the numbers of variables selected by the different methods across the bootstrap samples (Best-1 is not included because it always chooses exactly one variable). Lasso and elastic net choose a mode of twelve variables, with values from 1 to 20 distributed symmetrically. Reducing the mixing parameter, α, shifts the elastic net curve to the right, capturing more variables, but not increasing the accuracy of classification. While DLDA's number of variables is smaller than the first two, it has a longer tail, describing a few runs where over 20 variables were selected. Random forest tends to build more parsimonious models, but also sometimes will indicate that 20 or even 30 variables are useful in classifying samples.

4 Discussion

We have presented the application of five different methods of statistical classification to the same data set, embedding the training and evaluation of those methods in a bootstrap loop.

What is the effect of the bootstrapping? Bootstrapping has several beneficial functions. The first is to reduce the optimistic bias in the estimate of operating characteristics associated with the resubstitution estimates. While this could also be accomplished with an independent test set, once such a test set is used, it is no longer independent, and the risk of over-fitting the training set is transferred to the test set. A test set, if it is available, should be set aside until the cycle of model-fitting and evaluation is complete. Bootstrapping also facilitates an assessment of model robustness, and distribution-free estimates of variable importance and the variance of model coefficients.

What is the difference between the lasso and elastic net? These two methods are implemented in the same R function, but they differ only by changing the mixing parameter λ from 1 (lasso) to 0 (ridge regression). We used the value 0.9, based on a sensitivity analysis that is not shown. The main difference between their function is the way they handle correlated variables. Lasso will choose a single variable from a correlated cluster of variables, while elastic net will choose all of the correlated variables and assign them similar weights. This makes lasso better for constructing a parsimonious (but not necessarily unique) model, and elastic net better for identifying all the variables associated with a condition, but not for constructing the most parsimonious model. To demonstrate this, we simulated a data set of 40 variables on 100 observations, split evenly between two classes. The first five variables discriminated perfectly between the two classes, and had exactly the same value, while the other 35 variables were Gaussian random variables containing no information about the classes. Lasso chose the first variable, and no others. Elastic net chose the first five variables, and no other, and assigned them regression coefficient very nearly similar in value. Thus, lasso constructed a model that discriminated perfectly with the fewest variables possible, while elastic net

identified all the variables associated with the condition. Which of these analyses one would choose depends on one's objective. This experiment also demonstrates that the choice of variable in a correlated set may change considerably in response to small changes in the data, and may be arbitrary, or a function of irrelevant parameters (the first variable was chosen by lasso because of lexicographical ordering).

We believe that the results of modeling and discrimination tend to be over-interpreted. After a great deal of effort has been expended to determine the values of a great number of variables on a limited number of cases, there is a natural tendency to identify an effective discrimination function as a representation of "the truth." However, the likelihood of the probability of a specimen having the condition, depending on the variables, is laden with local maxima, and, no matter how systematic the search, there are almost certainly "better" models that are parsimonious or more accurate or both. This problem is exacerbated by the post-hoc nature of the variable quantification; some of the variables are defined after looking at the data, because they "look interesting." Inspecting a number of methods in a bootstrap cycle lends perspective to the training process by demonstrating the variation in the estimates of the discrimination model parameters, and also the variation in the variables selected for discrimination. The result should be a more nuanced appreciation of the limitation of a given model, selection of more appropriate discrimination methods and generation of hypotheses that can be tested in subsequent experiments.

5 R Code

5.1 Unzeroing, Stabilizing and Standardizing

```
#---uts.R

#---flow.csv is a comma-delimited ASCII file, where the first column is the class of
#   the specimen (NL,TUMOR) and the subsequent columns are the proportions of specific
#   types in various gates

flow.data.frame <- read.csv("flow.csv",sep=",")

y <- flow.data.frame$SAMPLE

#---Unzero using random 1/10 smallest value rule (Method 1)

v <- as.matrix(flow.data.frame[,2:ncol(flow.data.frame)])

#---Stabilize the values by identifying the smallest non-0 and largest non-1 values
#   for each variable, and using those values as parameters for a random uniform number
#   generator (runif)

min0 <- function(x){min(x[x>0])}; max1 <- function(x){max(x[x<1])}

min.v <- apply(v,2,min0); max.v <- apply(v,2,max1)

for (j in 1:ncol(v)) {
    n.0=sum(v[,j]==0)
    if (n.0>0) v[v[,j]==0,j] <- min.v[j]*runif(n.0)/10
    n.1=sum(v[,j]==1)
    if (n.1>0) v[v[,j]==1,j] <- 1-(1-max.v[j])*runif(n.1)/10
}

#---Apply the logit transform

x <- log(v/(1-v))

#---Standardize the columns of x

z <- x
for (j in 1:ncol(z)) z[,j] <- (z[,j]-mean(z[,j])/sd(z[,j]))
```

5.2 Heat Map

```
#---heatmap.R
library(ClassDiscovery)

source("uts.R")

cex <- 0.55

aspectHeatmap(z,Rowv=NA,labRow=y,col=redgreen(15),wExp=2.5,cexCol=cex,cexRow=cex)

```

The graphic in the paper is rotated by $90°$. Unlike the case of unsupervised learning, clustering in support of supervised learning groups the specimens together by their condition (`labRow=y`).

The `ClassDiscovery` package is available from the Biomathematics Department of M.D. Anderson. The function `heatmap` is available as part of the standard `R` distribution, but we prefer the `aspectHeatmap` function because it allows more control over the appearance of the graphic.

5.3 Cross-Validation

```
#---cross_validation.R

#---Bring in the data set for an example run; will generally not be used
#   in practice

source("uts.R")



n <- length(y)

#---10-fold Cross-Validation

K <- 10

cv.ind <- rep(NA,n)
for (i in 1:length(levels(y))) {
    n.i <- length(which(y==levels(y)[i]))
    cv.ind[which(y==levels(y)[i])] <- ((1:n.i)%%k)+1
}

cv.ind <- factor(cv.ind)

for (ind in 1:K) {

    z.out <- z[cv.ind==ind,]
    y.out <- y[cv.ind==ind]
    z.in <- z[cv.ind!=ind,]
    y.in <- y[cv.ind!=ind]

#---Train classifier here on z.in and y.in; use the classifier to
#   predict the condition y.out by applying the trained classifier to
#   z.out

}
```

This code does not perform the cross-validation, but sets up the K data sets on which cross-validation could be performed. The cross-validation code would be placed between the final comment and the final right bracket.

5.4 DLDA Classification

```
#---DLDA classification

#---Find alpha by cross-validation
find.alpha=function(x.std,y) {

    n=length(y); p=ncol(x.std)

#---calculate the single estimate of the standard deviation across
#   all variables
    n1=sum(y=="NL"); n2=sum(y=="TUMOR")
    m1=apply(x.std[y=="NL",],2,mean); m2=apply(x.std[y=="TUMOR",],2,mean)
    v1=apply(x.std[y=="NL",],2,var); v2=apply(x.std[y=="TUMOR",],2,var)
    sd.est=sqrt((n1*mean(v1)+n2*mean(v2))/(n1+n2))

#---set up the cross-validation loop for selecting the p-value.  The range
#   of possible alphas was determined by trial and error
    max.acc=0; cv.c=0
    cv.index=rep(1:10,ceiling(n/10)*10)[1:n]

    for (i.alpha in 7:60) {
        alpha=exp(-i.alpha/4)
        t=(m1-m2)/(sd.est/sqrt(n1+n2-2))

#-------select indexes the variables with extreme p-values
        select=as.numeric((pt(t,n1+n2-2)<alpha/2)|(pt(t,n1+n2-2)>1-alpha/2))

        if (sum(select)>0) {
            acc=0
            for (i.cv in 1:10) {
                train.1=x.std[(cv.index!=i.cv)&(y=="NL"),]
                train.2=x.std[(cv.index!=i.cv)&(y=="TUMOR"),]
                n.test.1=sum((cv.index==i.cv)&(y=="NL"))
                n.test.2=sum((cv.index==i.cv)&(y=="TUMOR"))
                test.1=matrix(x.std[(cv.index==i.cv)&(y=="NL"),],
                    n.test.1,p)
                test.2=matrix(x.std[(cv.index==i.cv)&(y=="TUMOR"),],
                    n.test.2,p)
                m1.cv=apply(train.1,2,mean)
                m2.cv=apply(train.2,2,mean)
                wx1=test.1%*%diag(select)%*%(m1.cv-m2.cv)/sd.est/sd.est-
                    as.numeric(t(m1.cv+m2.cv)%*%diag(select)%*%
                    (m1.cv-m2.cv)/2)/sd.est/sd.est
                wx2=test.2%*%diag(select)%*%(m1.cv-m2.cv)/sd.est/sd.est-
                    as.numeric(t(m1.cv+m2.cv)%*%diag(select)%*%
                    (m1.cv-m2.cv)/2)/sd.est/sd.est
                acc=acc+sum(wx1>cv.c)+sum(wx2<cv.c)
            }
            if (acc>=max.acc) {
                max.acc=acc
                opt.alpha=alpha
            }
        }
    }
    opt.alpha
}
```

```
predict.dlda=function(x.train,x.test,y.train,y.test,opt.alpha){
    n1=sum(y.train==levels(y.train)[1])
    n2=sum(y.train==levels(y.train)[2])

    m1=apply(x.train[y==levels(y.train)[1],],2,mean)
    m2=apply(x.train[y==levels(y.train)[2],],2,mean)

    v1=apply(x.train[y==levels(y.train)[1],],2,var)
    v2=apply(x.train[y==levels(y.train)[2],],2,var)

    sd.est=sqrt((n1*mean(v1)+n2*mean(v2))/(n1+n2))

    t=(m1-m2)/(sd.est/sqrt(n1+n2-2))
    select=as.numeric((pt(t,n1+n2-2)<opt.alpha/2)|(pt(t,n1+n2-2)>1-opt.alpha/2))
    if (sum(select)>0) {
        cv.c=0

        discrim.const=as.numeric(t(m1+m2)%*%diag(select)%*%
            ((m1-m2)/2))/sd.est/sd.est
        wx=(x.test%*%diag(select)%*%(m1-m2))-discrim.const

        y.predict=y.test
        y.predict[wx>cv.c]=levels(y.test)[1]
        y.predict[wx<=cv.c]=levels(y.test)[2]
        list(select,y.predict)
    }
}

m=apply(x.train,2,mean)
s=apply(x.train,2,sd)
x.train.std=(x.train-matrix(1,nrow(x.train),1)%*%m)/
    (matrix(1,nrow(x.train),1)%*%s)
x.test.std=(x.test-matrix(1,nrow(x.test),1)%*%m)/(matrix(1,nrow(x.test),1)%*%s)

opt.alpha=find.alpha(x.train.std,y)

#---sensitivity and specificity

ps=predict.dlda(x.train.std,x.test.std,y.train,y.test,opt.alpha)

boot.res[3,iboot,1]=sum(y.test==levels(y.test)[1])
boot.res[3,iboot,2]=sum(y.test==levels(y.test)[2])
boot.res[3,iboot,3]=sum((y.test==levels(y.test)[1])&
    (ps[[2]]==levels(y.test)[1]))
boot.res[3,iboot,4]=sum((y.test==levels(y.test)[2])&
    (ps[[2]]==levels(y.test)[2]))

#---variables selected and number of variables selected

x.selected[3,]=x.selected[3,]+ps[[1]]
nvar[3,sum(ps[[1]])]=nvar[3,sum(ps[[1]])]+1
x.ib.select[3,iboot,]=ps[[1]]

#---observations correctly classified

x.classified[3,bs3[y.test==ps[[2]]]]=x.classified[3,bs3[y.test==ps[[2]]]]+1

#---end DLDA classification
```

5.5 Recursive Partitioning Trees and Random Forest

```
#---random_forest.R
library(rpart)
library(randomForest)
source("uts.R")
zdf <- data.frame(y,z)

#---Build the additive formula with all predictors in Z
f <- formula(paste("y~",paste(colnames(z),collapse="+"),sep=""))

#---Construct the RPT and plot it
tree <- rpart(f,data=zdf)
plot(tree,margin=0.2)
text(tree, use.n=TRUE,pretty=0)

#---Random Forest
cut=c(n.train.nl,n.train.tumor)/(n.train.nl+n.train.tumor)
r=randomForest(x=x.train,y=y.train,xtest=x.test,ytest=y.test,ntree=1000,
    cutoff=cut,importance=TRUE)
```

5.6 Bootstrap with Elastic Net

```
#---boot.R
library(glmnet)
source("uts.R")

#---Determine the number of normal liver and tumor specimens
n.nl <- sum(y=="NL"); n.tumor <- sum(y=="TUMOR")

#---Determine the indices of the normal liver and tumor specimens
ind.nl <- which(y=="NL"); ind.tumor <- which(y=="TUMOR")

#---Number of variables
p <- ncol(z)

#---Number of bootstrap cycles
B <- 499; b <- 0

#---Initialize places to store results of bootstrap analyses
#   in.oob       The specimen is in the OOB
#   correct.oob  The specimen is correctly classified
#   x.selected   The variable is chosen

in.oob <- rep(0,n.nl+n.tumor)
correct.oob <- rep(0,n.nl+n.tumor)
x.selected <- rep(0,p)

#---The bootstrap loop
while (b<B) {

#---Get the bootstrap and OOB indices
    boot.ind.nl <- sample(x=ind.nl,size=n.nl,replace=TRUE)
    boot.ind.tumor <- sample(x=ind.tumor,size=n.tumor,replace=TRUE)
    boot.n.nl <- length(boot.ind.nl)
    boot.n.tumor <- length(boot.ind.tumor)

    oob.ind.nl <- setdiff(ind.nl,boot.ind.nl)
    oob.ind.tumor <- setdiff(ind.tumor,boot.ind.tumor)
    oob.n.nl <- length(oob.ind.nl)
    oob.n.tumor <- length(oob.ind.tumor)

#---Build the bootstrap and OOB analysis objects
    if ((oob.n.nl>0)&(oob.n.tumor>0)) {
        b <- b+1
        oob.y <- c(rep(0,length(oob.ind.nl)),rep(1,length(oob.ind.tumor)))
        oob.z <- rbind(z[oob.ind.nl,],z[oob.ind.tumor,])
        oob.ind <- c(oob.ind.nl,oob.ind.tumor)

        boot.y <- c(rep(0,n.nl),rep(1,n.tumor))
        boot.z <- rbind(z[boot.ind.nl,],z[boot.ind.tumor,])

#-------Elastic net classification
        enet.alpha=0.90
        glmnet1=cv.glmnet(boot.z,boot.y,family="binomial",alpha=enet.alpha)
        glmnet2=glmnet(boot.z,boot.y,family="binomial",lambda=glmnet1$lambda.min,
            alpha=enet.alpha)
        glmnet3=predict(glmnet2,oob.z,glmnet1$lambda.min,"response")

#-------Record the bootstrap results
        in.oob[oob.ind.nl] <- in.oob[oob.ind.nl]+1
        in.oob[oob.ind.tumor] <- in.oob[oob.ind.tumor]+1
        cut <- boot.n.tumor/(boot.n.tumor+boot.n.nl)
        correct.oob[oob.ind] <- correct.oob[oob.ind]+
            as.numeric((as.numeric(glmnet3>cut)==oob.y))
        x.selected <- x.selected + as.numeric(as.matrix(glmnet2$beta)!=0)
    }
}
```

This example demonstrates the use of the bootstrap with an elastic net classifier. The top part of the code (before `The bootstrap loop`, line 27) calculates specimen counts and initializes places to but results during the bootstrap loop. In the bootstrap loop, samples are drawn by randomly choosing vectors of indices (lines 29-38), which are then used to subset matrices of variables and vectors of specimen condition (lines 42-48). The actual classification is performed by only five lines (51-55); the results of classification are retained in 58-63. The matrices `in.oob, correct.oob` and `x.selected` would be processed after the loop to calculate sensitivity, specificity, accuracy and the variable importance index.

References

[1] R_Development_Core_Team. *R: A language and environment for statistical computing*. Vienna, Austria: R Foundation for Statistical Computing; 2008.

[2] Efron B. Bootstrap Methods: Another Look at the Jackknife. *Annals of Statistics* 1979;7:1-26.

[3] Fisher RA. The Use of Multiple Measurements in Taxonomic Problems. *Annals of Human Genetics* 1936;7:179-188.

[4] Dudoit S, Friedland J, and Speed TP. Comparison of discrimination methods for the classification of tumors suing gene expression data. *J Am Stat Assoc* 2002; 457:77-87.

[5] Breiman L. *Classification and regression trees*. Belmont, Calif.: Wadsworth International Group; 1984. 358 p. p.

[6] Breiman L. Random Forests. *Machine Learning* 2001;45:5-32.

[7] Tibshirani R. Regression Shrinkage and Selection via the Lasso. *J. Royal Statist. Soc. B* 1996;58:267-288.

[8] Zou H, Hastie T. Regularization and variable selection via the elastic net. *J. Royal Statist. Soc. B* 2005;67:301-320.

[9] Friedman J, Hastie T, Tibshirani R. Regularization Paths for Generalized Linear Models via Coordinate Descent. *J Stat Softw* 2010;33:1-22.