2006

Quantifying the Effects of Correlated Covariates on Variable Importance Estimates from Random Forests

Ryan Vincent Kimes
Virginia Commonwealth University

QUANTIFYING THE EFFECTS OF CORRELATED COVARIATES ON VARIABLE IMPORTANCE ESTIMATES FROM RANDOM FORESTS

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Science at Virginia Commonwealth University.

by

RYAN VINCENT KIMES
Bachelor of Science in Statistics, California Polytechnic State University San Luis Obispo, 2004

Director: KELLIE J. ARCHER, PH.D.
ASSISTANT PROFESSOR, DEPARTMENT OF BIOSTATISTICS

Virginia Commonwealth University
Richmond, Virginia
May 2006

Acknowledgement

I would like to thank my thesis advisor Dr. Archer for all the help she provided with my programming, analysis, and for sparking my interest in the field of gene expression analysis. I really appreciated her availability and extremely fast edit checking. I would also like to thank Dr. Ramesh and Dr. Mas for being on my committee. I would like to thank the entire Biostatistics department for being so supportive and for providing such a great atmosphere for education. I would also like to acknowledge the students who were always so supportive of each other, especially my first year officemates. I couldn't have done this without the support of my parents Paul and Helen Kimes, and all my family and friends.

Table of Contents

Page

List of Tables

Page

List of Figures

Page

Abstract

QUANTIFYING THE EFFECTS OF CORRELATED COVARIATES ON VARIABLE IMPORTANCE ESTIMATES FROM RANDOM FORESTS

By Ryan Vincent Kimes

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Science at Virginia Commonwealth University.

Virginia Commonwealth University, 2006

Major Director: Kellie J. Archer, Ph.D.
Assistant Professor, Department of Biostatistics

Recent advances in computing technology have lead to the development of algorithmic modeling techniques. These methods can be used to analyze data which are difficult to analyze using traditional statistical models. This study examined the effectiveness of variable importance estimates from the random forest algorithm in identifying the true predictor among a large number of candidate predictors. A simulation study was conducted using twenty different levels of association among the independent variables and seven different levels of association between the true predictor and the

response. We conclude that the random forest method is an effective classification tool when the goals of a study are to produce an accurate classifier and to provide insight regarding the discriminative ability of individual predictor variables. These goals are common in gene expression analysis, therefore we apply the random forest method for the purpose of estimating variable importance on a microarray data set.

CHAPTER 1 Background

1.1 Two Cultures

In the field of statistics, there are two methodological cultures statisticians adhere to when drawing conclusions from data. The dominant culture is that of statistical modeling. Statistical models are developed using mathematical theory and making distributional assumptions. The questionable nature of statistical models is whether we can really assume that natural phenomenon follow some specified distribution that can be described solely by its mean and variance. While the modeling approach can be effective, often it ignores reality; nature does not operate by parsimony nor abide by underlying model and distributional assumptions.

As computing power has increased, a second statistical culture has emerged. Grown from the field of machine learning, algorithmic modeling has developed as an alternative to statistical modeling. Algorithmic models can answer the same statistical questions such as identifying important covariates and making inferences regarding responses, but does so with minimal assumptions and less focus on "model building" or parsimony. In analyzing any given data set, a statistician's focus should always be on identifying the best solution, whether this comes from a statistical or algorithmic model. Moreover, the strategy for any data analysis task should be selected based on criteria such as predictive accuracy.

The data we observe from nature come from a mysterious and highly complex black-box. In the statistical modeling culture, data are assumed to arise from a specified distribution and therefore conclusions are often about the model's mechanism rather than nature's mechanism, which is what we are attempting to study. Therefore parametric models imposed on data originating from such complex systems result in a loss of accuracy and information. Models that best emulate nature in terms of predictive accuracy are the most complex and are difficult to dissect. Breiman (2001b) suggests that the statistician should not be deciding between accuracy and interpretability, but instead should be focusing on obtaining useful information. Algorithmic models focus on the strength of predictors, convergence, and good predictive accuracy. The only assumption made in algorithmic modeling is that the data are drawn independent and identically distributed (i.i.d) from some unknown multivariate distribution.

Breiman's (2001b) focus on algorithmic models led to the development of the random forest methodology. Two key ideas in statistics form the basis for random forests, classification and regression trees (CART) and bootstrap aggregation (Breiman 1996). These topics are introduced in the following two sections.

1.2 Classification Trees

Classification methods are used to find a systematic method for predicting the class of an observation based on a given set of measurements. Binary tree structured classifiers, commonly known as classification trees, have been an intuitive method for describing relationships in data (Breiman 1998). The purpose of a classification tree is to

produce an accurate classifier that uncovers the predictive structure and nature of the data. The classifier is created by repeatedly splitting the data into two descendent subsets based on optimal cutpoints chosen for variables in the predictor space. A new observation is classified by following the decisions at each node down the tree until it reaches a terminal node. The predicted class for the terminal node is taken to be the predicted class of the new observation. More formally, suppose the data consist of n observations denoted $\mathcal{L}=\{(\omega_1,\mathbf{x}_1),(\omega_2,\mathbf{x}_2),...,(\omega_n,\mathbf{x}_n)\}$ where ω_i is one of $j=1,...,J$ classes and $\mathbf{x}_i$ is a vector of p covariates. All n observations start together in what is called the root node.

The classification tree algorithm proceeds as follows:

1) For node t, find the best split s for each of the p independent variables. There will be p best splits.
2) Of the p best splits s, select s^*, the best of the best splits. This variable and the identified cutpoint c_{s^*} is used as the primary split for the node.
3) From the remaining variables find the k splits that are most similar to s^*. These will be the surrogate splits for the node.
4) Split the data at the node by sending all observations with $x_i \leq c_{s^*}$ to the left descendent node and all observations with $x_i > c_{s^*}$ to the right descendent node.
5) Continue steps 1-4 for all subsequent nodes of the tree until a stopping rule is achieved.

The idea is to select each split of a subset so that the data in the descendent subsets are more homogeneous with respect to class than the data in the parent node.

To determine the best split:

1) Define the node proportions in node t to be $p(\omega_j \mid t)$ where $j = 1, \ldots, J$. This is the proportion of cases in node t belonging to class ω_j, so

   $p(\omega_1 \mid t) + \ldots + p(\omega_J \mid t) = 1$.

2) The impurity function ϕ is defined as a non-negative function of the proportions $\left(p(\omega_1 \mid t), \ldots, p(\omega_J \mid t)\right)$. ϕ can be any function that has the following properties: ϕ takes on a maximum when all classes are mixed equally together and a minimum when the node is composed of one class. Symbolically,

   $$\phi\left(\frac{1}{J}, \frac{1}{J}, \ldots, \frac{1}{J}\right) = \text{maximum and}$$

   $$\phi(1,0,0,0,\ldots,0) = \phi(0,1,0,0,\ldots,0) = \ldots = \phi(0,0,0,0,\ldots,1) = 0.$$

3) The impurity measure at each node is defined as $i(t) = \phi$.

4) Consider a candidate split s that will split node t into left and right descendent nodes, t_L and t_R. Choose s to maximize the decrease in node impurity, defined as

   $$\Delta i(s,t) = i(t) - p_L i(t_L) - p_R i(t_R)$$

   where p_L and p_R are the proportions of observations in t_L and t_R respectively.

5) The best split s^* out of S possible splits is defined as: $\Delta i(s^*,t) = \max_{s \in S}(\Delta i(s,t))$.

For observations having missing values for the best splitting variable, surrogate splits are used for determining descendent node assignment. Surrogate splits are also used in the estimation of variable importance which will be described later.

The Gini criterion is the most commonly used impurity function. Breiman (1998) cites the Gini criterion as the preferred method to split classification trees, having the form: $i(t) = \sum_{j \neq i} p(\omega_j \mid t) p(\omega_i \mid t)$.

The stopping rule is chosen to minimize bias and predictive accuracy. If the tree is grown too large, it will be biased and overfit the data. That is, growing an overly large tree is analogous to including increasingly more predictors in a linear model. On the other hand, if the tree is not large enough, too much error will be induced. A balance between bias and error can be achieved by either using a strict stopping rule or by growing a full size tree, with no stopping rule, then pruning. A stopping rule designates a terminal node when no significant decrease in node impurity is possible. The stopping rule basically grows a tree until it reaches an acceptable purity level. Pruning works conversely. The tree is grown until all terminal nodes are 100% homogenous, some of which may contain only one observation. Nodes of the tree are then pruned off until a desired level of complexity is achieved, using, for example, the 1-SE rule (Breiman 1998). As recommended by Breiman (2001a), the classification trees in Random Forests will be grown to full size without any stopping rule or pruning.

Classification trees can be very useful. They are easy to interpret and simple to apply. They work efficiently for large data sets and can handle a large number of predictors, both continuous and categorical. They feature automatic variable selection for

complexity reduction and generalization error can be estimated using cross-validation. The disadvantages of classification trees is that they can have somewhat high error rates and are not robust. Classification trees are very sensitive to small changes to the data in the learning sample.

1.3 Bootstrap Aggregating

When modeling any sort of data the statistician always runs into a tradeoff between variance and bias. If a classification tree is grown to full size, its prediction error will be very low for the sample used to train, but it will be extremely biased. The tree will not be useful in making predictions on a new set of data from the same population. If a classification tree is grown to be very small, its prediction error will be high, but bias will be low. Theoretical knowledge of the data and subject matter information can help in choosing the correct model. If a large dataset is available, it can be partitioned into training and test datasets. The model is derived using the training data, and error is assessed using the test data. If a large sample is not available, cross-validation can also be used estimate the error rate. It would be of more help to have knowledge of the distribution that produced the learning set. Although this distribution is usually not known, it can be imitated. That is, new quasi-samples from the empirical distribution can be created by sampling with replacement from the original learning set. This is called bootstrapping (Hastie 2001). Unstable procedures such as classification trees can be stabilized by acquiring bootstrap samples of the learning dataset, growing trees on each of the bootstrapped samples, and then averaging over the tree predictors. The process of

aggregating classifiers over multiple bootstrap resamples is called bootstrap aggregating, or bagging for brevity.

The bagging classification tree algorithm is as follows:

1) Take a sample of size n with replacement from the original dataset $\mathcal{L}$. Call this bootstrapped sample $\mathcal{L}_b$.
2) Grow a classification tree using $\mathcal{L}_b$.
3) Prune the tree using the original data set $\mathcal{L}$. Save the predicted class for each observation.
4) Repeat steps 1-3 B times.

The bagged predicted class is the class with the majority vote from the B trees. It is important to note the difference in the learning and test sets for bagged trees compared to ordinary classification trees. Classification trees are usually grown on a portion of the original sample, called the learning set. The rest of the sample is used as the test set. In bootstrap aggregation the learning sets are bootstrapped samples of the original data. The test set is the original data. The set of bagged trees has a much lower error rate than the ordinary classification tree (Breiman 1996). Opitz and Maclin (1999) compared bagging to a single tree classifier on 23 datasets. They found that bagging performed better than a single classifier in almost all cases. The disadvantage of bagging is the loss of interpretability - a bagged tree is no longer a tree.

1.4 Random Forests

Breiman (2001a) developed an extension of bagging classification trees, called random forests. Random forests are a special instance of bagging classification trees but with the additional characteristics of random feature selection at each node and no pruning or stopping rule. A group, or forest, of trees is grown and the aggregation of them is taken to be the classifier. Again, each tree is grown using a bootstrapped sample $\mathcal{L}_b$ from the original learning sample $\mathcal{L}$. The difference is that at each node of the tree, m of the independent variables are randomly selected from which to choose to split. The random selection of features at each node decreases the correlation between the trees in the forest thus decreasing the forest error rate. The smaller the value of m the less correlated the trees in the forest and the smaller the forest error rate. However m cannot be too small because as m decreases, the strength of the trees decreases. This is the only parameter to adjust in random forests.

The random forest algorithm is as follows:

1) Sample N observations with replacement from the learning dataset $\mathcal{L}$. Call this the Bootstrap Sample $\mathcal{L}_b$.
2) For node t, take a sample of size m from the p independent variables, usually $(m = \sqrt{p})$.
3) Find the best split s for each of these m variables using the observations in $\mathcal{L}_b$.

4) Choose the best of the *m* best splits to split the node, denote s*. This variable and the identified cutpoint c_{s*} is used to split the node.
5) Split the data at this node by sending all observations with $x_i < c_{s*}$ to the left descendent node and all observations with $x_i \geq c_{s*}$ to the right descendent.
6) Repeat steps 2-5 procedure to grow a maximally sized tree.
7) Repeat steps 1-6 *B* times.

The Gini criterion is used to select the split with the lowest impurity at each node. For each tree in the forest, the predicted class for each observation is saved. The class with the maximum number of votes among the *B* trees in the forest is the predicted class of an observation.

There are two extremely useful byproducts of random forests, out-of-bag observations and variable importance measures. Since each tree is grown from a bootstrapped sample $\mathcal{L}_b$, on average, about one-third of the observations in the data set will not be used to grow the tree. Each bootstrap sample $\mathcal{L}_b$ is sampled with replacement from the data. All observations in the learning sample $\mathcal{L}$ have an equal probability of being selected every time a new observation is added to $\mathcal{L}_b$. The probability that an observation is selected at least once is $1-(1-\frac{1}{N})^N$. For a large *N* this is approximately $1-\frac{1}{e} \approx 0.632$. Therefore only two-thirds of the observations in $\mathcal{L}$ will be used to build a tree on $\mathcal{L}_b$ .The rest of the observations are considered the out-of-bag (oob) observations

for that tree. Each tree will vary with respect to the oob observations. These oob observations form a natural test set for each tree, rather than using the computational expensive cross validation method to estimate the error of the random forest. The oob observations will also be used to calculate variable importance.

Random forests have a number of advantages over other machine learning methods. They have been shown to have some of the best accuracy among current algorithms (Breiman 2001b; Diaz-Uriarte and Alvarez de Andres 2006). Breiman (2001b) compared the performance of single trees to random forests on a number of data sets. For most of the data sets the misclassification error was reduced by at least one-third and sometimes one-half for the forests compared to the trees. The Statlog Project (Breiman 2001b) compared 18 different classifiers including neural nets, CART, linear and quadratic discriminant analysis, nearest neighbor, and others. There were four data sets in the project that came with separate test sets. Random forests ranked number 1 in accuracy on all four data sets, the next best method averaged a rank of 7.3 on the data sets. They run efficiently on large data sets and can handle thousands of input variables. They also generate variable importance measures and an internal estimate of error without cross-validation. They are not computationally intensive and can be saved for future use on other data.

Random forests have the same drawbacks as bootstrap aggregation. The average person can look at a decision tree and understand the model, since the tree provides a visible structure illustrating the decisions made at each node on the tree, which can be easily interpreted. Random forests consist of B trees, where B is customarily at least 500.

One cannot synthesize the information presented in 500 trees. Moreover, the predicted class is the class having a maximum vote among the B trees making the predictive mechanism difficult to uncover. Therefore, the decisions are made in a "black-box," similar to the way that data are generated in nature. The predictive accuracy of random forests have been demonstrated to be very good, but it becomes difficult to render an interpretation.

1.5 Variable Importance (Tree and Forest)

In situations where there are a large number of candidate predictors and a small number of observations, it becomes difficult to determine which variables are most important for accurate prediction. There are a number of statistical modeling approaches that can be used to find a better subset of the independent variables. Such approaches are often not estimatable when the number of covariates p is much larger than the sample size N, $p > N$. Moreover, most statistical modeling approaches assume independence among the p candidate predictors. With large p, the analyst will likely find a multiplicity of models that can achieve a similar level of prediction error. Each model will provide a different interpretation of the underlying data generating mechanism. Situations where $p > N$ call for an algorithmic approach to identify the important predictor variables.

Variable importance can be calculated using classification trees. To motivate the method we must keep in mind that there will be situations where a variable x_j is never chosen as a primary split in the classification tree, but when another variable x_i is removed from the tree x_j suddenly becomes the primary variable on which to split. This

motivates the use of surrogate splits in the calculation of variable importance. The sum of the change in Gini node impurity for all splits in the tree where x_m is a primary or surrogate split is taken as an estimate of the importance of variable x_m. When this sum is large, x_m is interpreted as being an important variable in reducing node impurity and thus is an important predictor. When this sum is small, x_m is interpreted not to substantially reduce node impurity; therefore, it is not an important predictor. It should be noted that the number of surrogate splits k that are saved at each primary node is a user-specified parameter. Therefore, choice of k may have some effect on variable importance estimates. The disadvantage of using classification trees to calculate variable importance is that small changes in a sample can drastically change which variables get selected to split the node, leading to false impressions based on variable importance estimates. Classification trees also run into the "multiplicity of good models" dilemma previously mentioned. With a large number of predictor variables, two different trees can be created that have almost the same prediction error but give two entirely different interpretations of the nature of the data.

The use of surrogate splits for estimating variable importance is not plausible under the random forest framework since only m of the p predictors are examined for splitting at each node. However, random forests provide two robust measures of variable importance. The first measure of variable importance is based on the Gini criterion. This is calculated similarly to the classification tree variable importance. Specifically, at each split the decrease in the Gini node impurity is recorded for the variable x_i that is split on. The average of all decreases in the Gini impurity in the forest where x_i is used to split is

used to get the Gini importance estimate. Another method of variable importance uses the mean decrease in accuracy. To estimate variable importance using mean decrease in accuracy:

For independent variables $i=1,...,p$:

1) For the b^{th} tree in the random forest, identify the oob observations, $\mathcal{L}_{oob,b}=\mathcal{L}-\mathcal{L}_b$.
2) Put the oob cases down the tree and sum the number of times the tree predicts the correct class.
3) Next, "mess up" the values of the independent variable x_i by randomly permuting them in the $\mathcal{L}_{oob,b}$ sample.
4) Put the "messed up" oob cases down the tree and sum the number of times the tree predicts the correct class.
5) Subtract the number of votes for the correct class in "messed up" oob data from the number of votes for the correct class in the untouched oob data.
6) Repeat steps 1-5 for $b=1,...,B$. The average over the B trees is the importance score for variable x_i.

The mean decrease in accuracy importance scores are standardized. A large importance score indicates that a variable is commonly selected to be included in the trees in the forest and that generally the nodes that split on this variable are important for classification. Breiman (2004) recommends running a forest consisting of at least 2500 trees when variable importance measures are estimated.

Variable importance measures are helpful for studying the mechanism that has produced the data. Rather than estimate a specific relationship between the independent variables and the response as in data modeling, the variable importance measures give robust estimates of which variables are most important to the random forest's emulation of the natural mechanism behind the data.

A number of papers have been written to show that random forests are effective classifiers, but very few have studied variable importance measures (Breiman 2001a; Breiman 2001b; Svetnik et al. 2003; Diaz-Uriarte and Alvarez de Andres 2006). The focus of this thesis is to study how well random forests can identify the actual class predictor among a large number of possible independent variables using variable importance measures. We are specifically interested to see how well the actual predictor can be identified when a number of independent variables are correlated with it. An extensive simulation study was used to examine how well random forests identify the correct predictor variable with different levels of correlation (ρ) among the independent variables and different levels of association (β) between the actual predictor and the response. The simulation study is described in chapter 2 and the results of the simulation study are reported in section 2.2. Moreover, since microarray studies commonly produce datasets consisting of a large number of candidate predictors and a small number of observations, the random forest methodology is demonstrated on a microarray dataset presented in Chapter 3.

CHAPTER 2 Simulation Study

2.1 Simulation Study Description

Simulation was used to examine how well random forests identify the correct predictor variable with different levels of correlation (ρ) among the independent variables and different levels of association (β) between the actual predictor and the response.

One hundred observations having 800 covariates and one response class were simulated to resemble observations from a microarray study. A set of independent variables were randomly generated using randomly selected means ranging from 6 and 12. This is a typical range for $\log_2$ transformed gene expression values measured on an absolute intensity scale. All variables were generated as having a variance of 1. The random variables were created in sets of 40, drawn from a multivariate normal distribution having specific internal-set correlations. For example, x_1 through x_{40} were generated as having a correlation of 0.00, x_{41} through x_{80} were generated as having a correlation of 0.05, x_{81} through x_{120} were generated as having a correlation of 0.10,..., x_{761} through x_{800} were generated as having a correlation of 0.95. Correlations range from 0.00 to 0.95 by increments of 0.05. The data set contains 20 different correlations with 40 variables per correlation for a total of 800 independent variables.

Thus for $i = 1,\ldots,20$ the vectors $\mathbf{x}_{40(i-1)+1},\ldots,\mathbf{x}_{40(i-1)+40}$ are each random samples of size $N = 100$ where $\mathbf{x}_{40(i-1)+1},\ldots,\mathbf{x}_{40(i-1)+40} \sim N_{40}(\mu_i, \Sigma_i)$. μ_i is a vector of length 40 randomly drawn from $U[6,12]$ and

$$\Sigma_i = \begin{bmatrix} 1 & \rho_i & \rho_i & . & \rho_i \\ \rho_i & 1 & \rho_i & . & \rho_i \\ \rho_i & \rho_i & . & . & . \\ . & . & . & . & . \\ \rho_i & \rho_i & . & . & 1 \end{bmatrix}_{40X40}, \text{ where } \rho_i = \frac{i-1}{20} \text{ and } \Sigma_{ij} = \mathbf{0}_{40X40} \text{ for } i \neq j.$$

The dichotomous response was artificially generated using one of the independent variables, $x_1,\ldots,x_{800}$. In each simulation, one variable of the 800 independent variables x_i was chosen to be the actual predictor of the response. To generate the n=1,…,100 dichotomous responses, the independent covariates were first mean centered and scaled. Then the probability of observation n in x_i belonging to class 1, $\pi(x_{i,n})$ was calculated using these standardized observations and the chosen value of β as

$$\pi(x_{i,n}) = \frac{e^{\beta x_{i,n}}}{1+e^{\beta x_{i,n}}}.$$

Then, one hundred observations were generated from a uniform distribution on the interval [0, 1], denote c_n. If $\pi(x_{i,n})$ was greater than its matching cutoff value then the observation was assigned to class 1, otherwise, it was assigned to class 0. Formally, the response y_n was taken to be

$$y_n = \begin{cases} 1 & \pi(x_{i,n}) > c_n \\ 0 & \pi(x_{i,n}) \leq c_n \end{cases}.$$

This generated a classification response y_n that has a specified relationship to the actual predictor based on the value of β, but with some random noise induced by the randomly uniform generated values. A drawback of this method is that a logistic regression would actually provide the best model for the data. However, as with microarray data, it is not feasible to fit a multiple logistic regression model using 800 predictors and 100 observations, especially in the presence of high collinearity. Also, there is no obvious way to generate responses based on the modified classification trees found in random forests.

One of the 800 independent variables is chosen to be the actual predictor of the response for each simulation. A total of 140 simulations were done, one for each correlation value (0.00 to 0.95 by 0.05) and value of β (0.25, 0.50, 0.75, 1.00, 1.25, 1.50, 1.75). The β values correspond to a range of odds ratios from approximately 1.28 to 5.76. For each correlated set, the first variable in the set of 40 was used to generate the response variable, or $\mathbf{x}_{40(i-1)+1}$. For example, when studying a correlation of 0.00, x_1 is the actual predictor of the response. When studying a correlation of 0.05, the actual predictor is x_{41}. The variable importance measures for each of the 800 variables were estimated along with the out-of-bag error for each forest.

Simulation Algorithm:

1) Generate response variable y_n using $x_{i,n}$ as the actual predictor, and the strength of the relationship between $x_{i,n}$ and the response is β.
2) Build a large random forest of size $B = 2500$.

3) Estimate the variable importance for all the independent variables and record the out-of-bag error for the forest.
4) Repeat steps 1 through 3, building R =100 random forests. There will be 100 random samples with the same predictor and value of β. Each variable will now have 100 variable importance estimates. There will also be 100 measurements of out-of-bag error.
5) For each variable, take the average of variable importance estimates over the R =100 RFs. Each variable will now have an average simulation variable importance. Also, average the out-of-bag errors for the 100 RFs.
6) Repeat steps 1 through 5 with different values of β but with $x_{i,n}$ remaining as the actual predictor.
7) Now repeat steps 1 through 6 but replace $x_{i,n}$ with the predictor from the next correlation set.
8) Continue this process until predictors from each correlation set have been used with all values of β.

Simulations were conducted using the R programming environment (R Development Core Team 2005) (see Appendix B). Software to implement the random forest algorithm was originally written in Fortran by Breiman (2004). The randomForest library for R was created by Liaw and Wiener (2002) based on the original Fortran code. The randomForest library was used with all default settings except that 2500 trees were specified to be grown. The value for m, the number of independent variables to choose

from at each node, was set to the default of $m = \sqrt{p}$. Breiman (2004) mentions this as the ideal value to use for classification. We note that other researchers have found RFs can be insensitive to the choice of m. For example, Svetnik et al. (2003) used 5-fold cross-validation to assess the random forest error rate over a range of values of m. They found that the performance of random forests change very little over a large range of values of m, except when m is extremely small or large, $m = 1$ or p. Diaz-Uriarte and Alvarez de Andres (2006) examined out-of-bag error using different values of m for nine microarray data sets. They found the out-of-bag error to be relatively stable and mention the default setting as a good choice for m. Therefore, the use of $m = \sqrt{p}$ is reasonable. The mean decrease in accuracy importance scores and the Gini importance scores were saved for each random forest in the simulation study for further examination.

To provide a comparison between random forests and a standard data modeling technique, logistic regression was used to try to identify the actual predictor in the correlated set. Response variables were generated for $\beta = 1.75$ and $\rho = 0.95$ using the first variable in this correlated set from the simulation data set (x_{761}). Univariable logistic regression models were fit to the response using each of the 40 variables in the correlated set $x_{761},\ldots,x_{800}$ as the independent variables. The likelihood ratio statistics were calculated for each of the 40 models. To get a measure of importance, the likelihood ratio statistics were rank ordered. This was repeated 100 times and the proportion of times that the actual predictor was ranked first was recorded. In addition, we attempted to fit logistic regression models including all 40 variables in the correlated set, to determine if the algorithm could converge in the presence of high degree of co-linearity.

2.2 Simulation Study Results

The average variable importance for all 800 variables for selected values of β {β = (0.25, 1.00, 1.75) } and ρ {ρ = (0.20, 0.40, 0.60, 0.80, 0.95) } are presented in Figures 1a-1c. These figures depict, for each simulation (that is, combination of β and ρ), the average variable importance estimate for the actual predictor displayed as a square, the average variable importance for those variables within the same correlated set as the true predictor displayed as open circles, and all other variables displayed as gray points. Clearly for β=0.25 the variable importance for the actual predictor is not clearly distinguishable for the remaining points. However, in all other cases the variable importance estimate for the actual predictor is clearly distinguished from those points not in its correlated set. For clarity, the remaining plots exclude the variable importance estimates for those variables that are not in the actual predictor's correlated set.

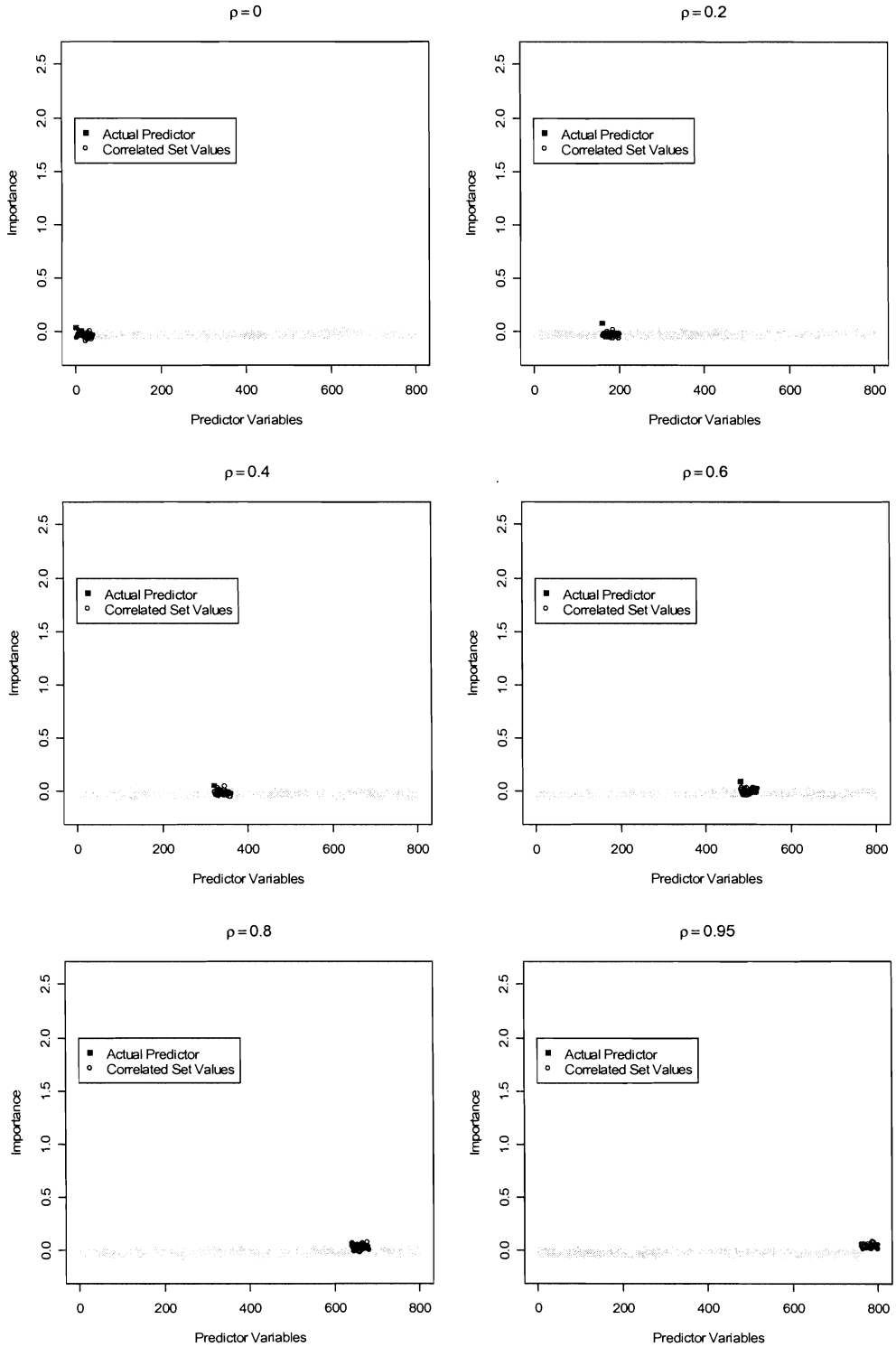

Figure 1a: Plot of the average variable importance for each of the 800 predictors when β = 0.25 and ρ = (i) 0; (ii) 0.20; (iii) 0.40; (iv) 0.60; (v) 0.80; and (vi) 0.95

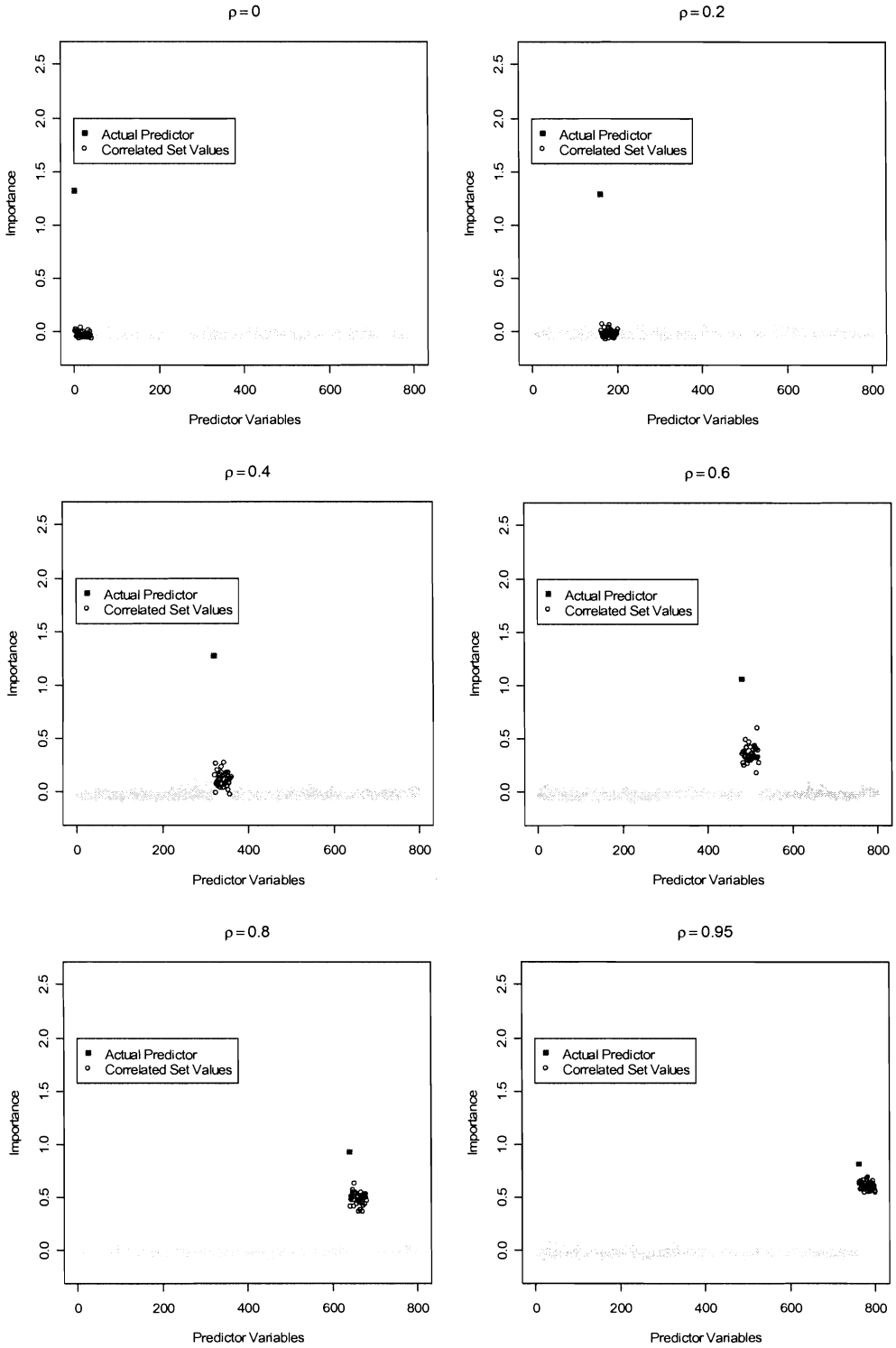

Figure 1b: Plot of the average variable importance for each of the 800 predictors when β = 1.00 and ρ = (i) 0; (ii) 0.20; (iii) 0.40; (iv) 0.60; (v) 0.80; and (vi) 0.95

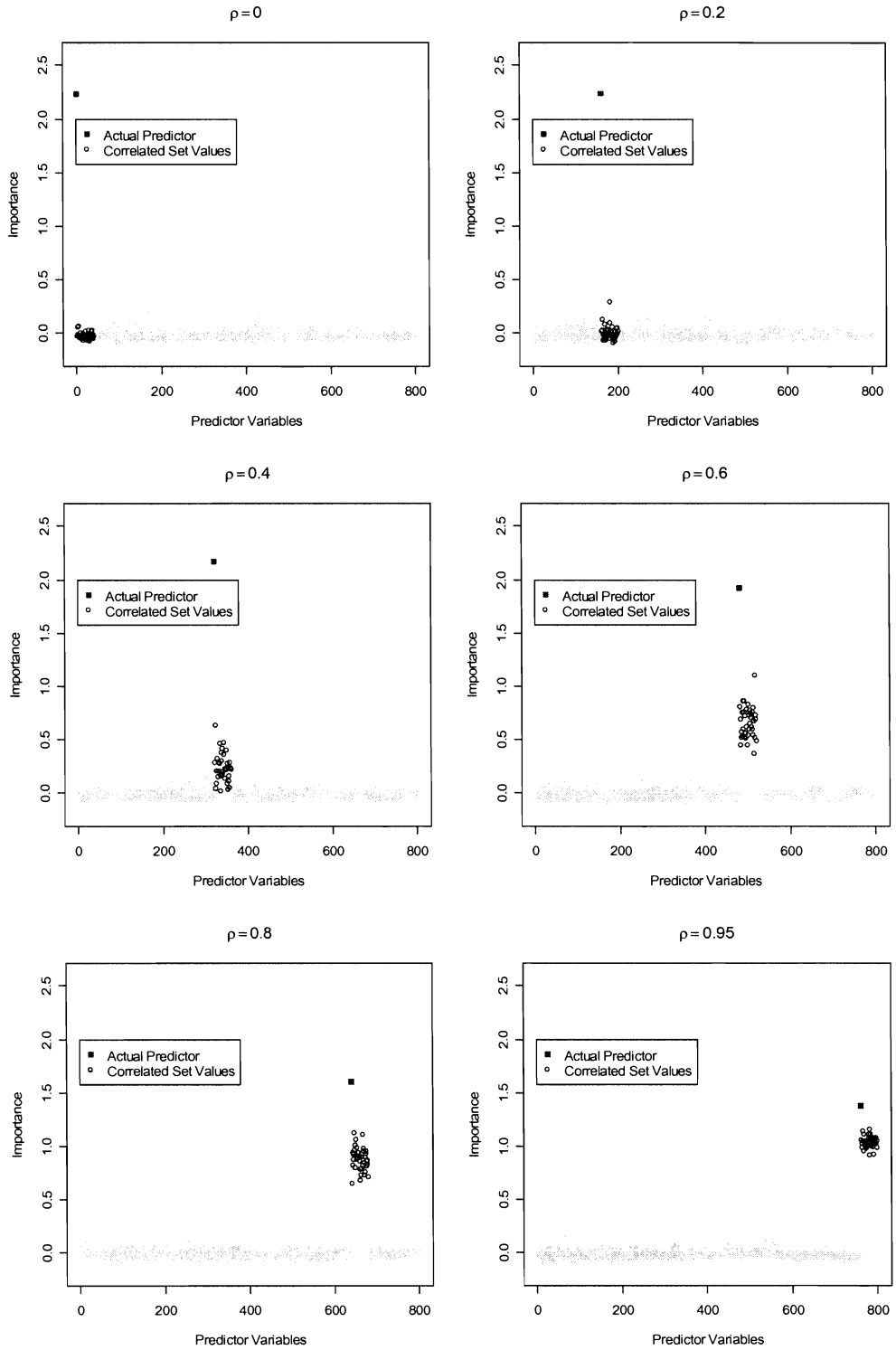

Figure 1c: Plot of the average variable importance for each of the 800 predictors when β = 1.75 and ρ = (i) 0; (ii) 0.20; (iii) 0.40; (iv) 0.60; (v) 0.80; and (vi) 0.95

A more concise summary of the entire simulation study is presented graphically in Figures 2a-2b. Each graph in these figures depicts a set of 20 simulations for a specific value of β and for all correlation values. The average variable importance estimate for the actual predictor is displayed as a square, the average variable importance for those variables within the same correlated set as the true predictor are displayed as open circles, the points are jittered so that the distribution of the points in the correlated set can be seen, and lines connect the means of the sets. When $\beta = 0.25$ the actual predictor cannot be identified among the other correlated independent variables. There is no separation between the actual predictor variable importance estimates and the remaining variable importance estimates. When $\beta = 0.50$ the actual predictor can be identified even when the set correlation is as high as 0.75. When $\beta = 0.75$ the actual predictor can be identified in all instances except when the correlation is 0.90 and 0.95. For β values greater than 0.75, the actual predictor can be identified in all simulations for all correlation values.

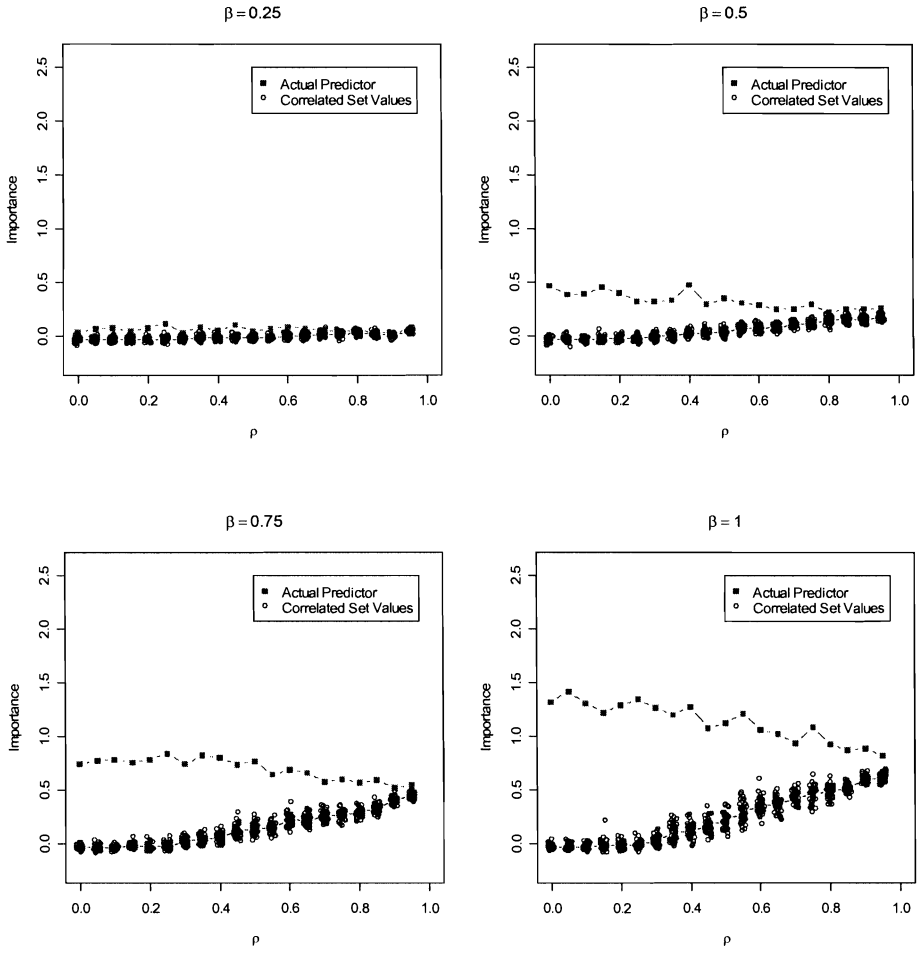

Figure 2a: Plot of the average variable importance for the actual predictor and the variables in the correlated set at all values of ρ when β = (i) 0.25; (ii) 0.50; (iii) 0.75; (iv) 1.00

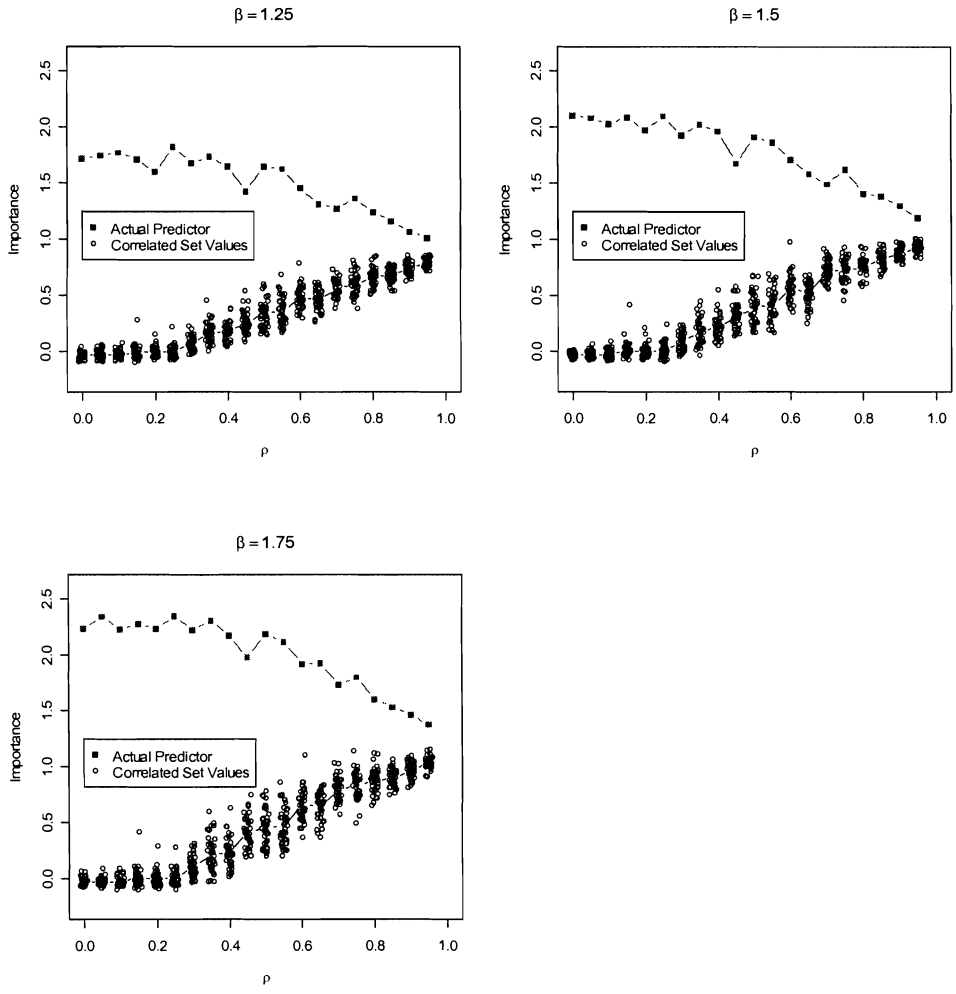

Figure 2b: Plot of the average variable importance for the actual predictor and the variables in the correlated set at all values of ρ when β = (i) 1.25; (ii) 1.50; (iii) 1.75

There are a few general trends in the simulations. The graphs in Figures 3a-3d use the same formatting as Figures 2a-2b except that they depict sets of 7 simulations for specific values of ρ and for all β. Looking at Figures 3a-3d we see that as the value of β increases, the variable importance estimates for the actual predictor increases and the variable importance estimates for the variables in the set that are correlated with the predictor also increases.

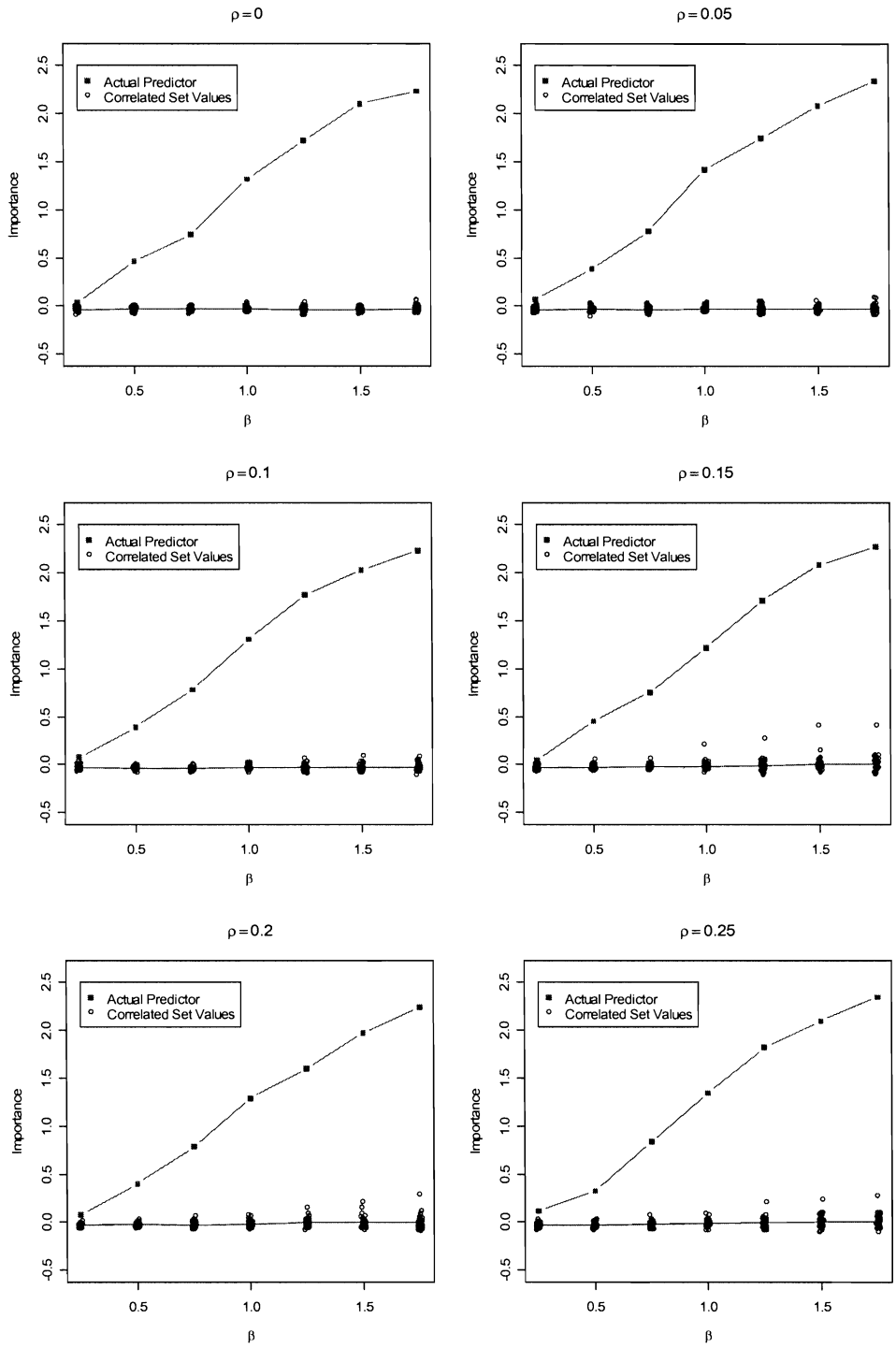

Figure 3a: Plot of the average variable importance for the actual predictor and the variables in the correlated set at all values of β when ρ = (i) 0.00; (ii) 0.05; (iii) 0.10; (iv) 0.15; (v) 0.20; (vi) 0.25

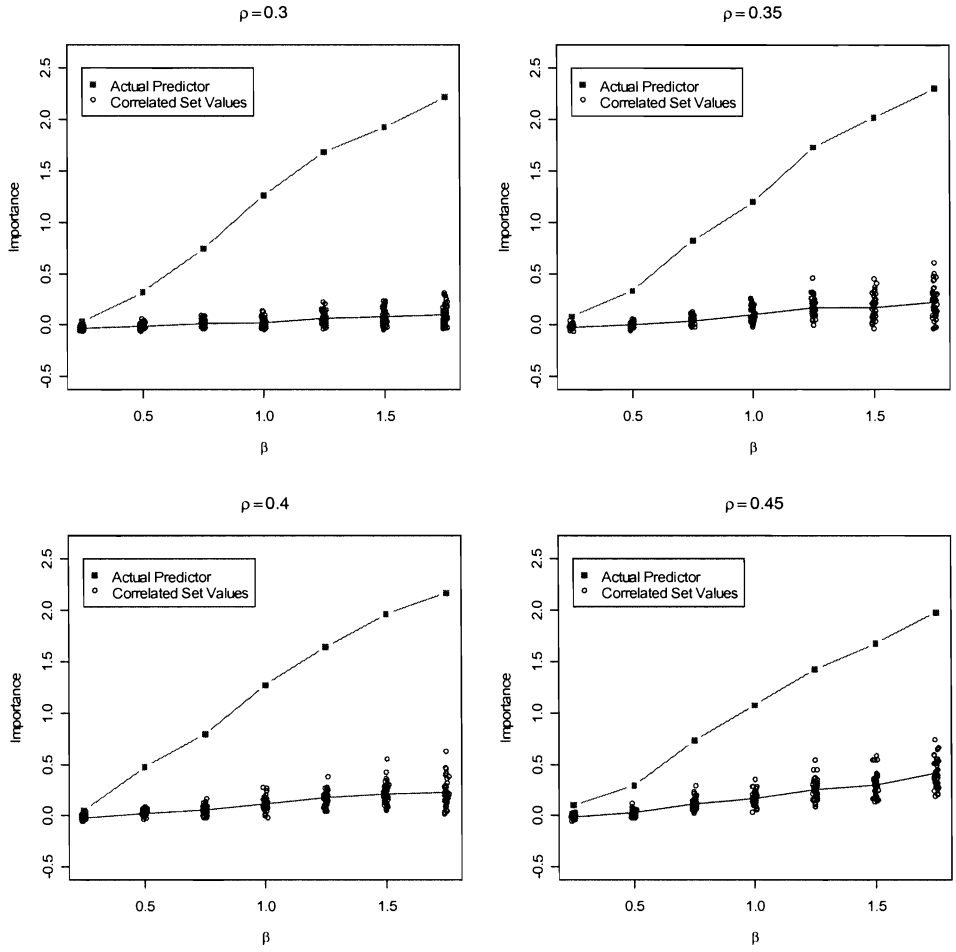

Figure 3b: Plot of the average variable importance for the actual predictor and the variables in the correlated set at all values of β when ρ = (i) 0.30; (ii) 0.35; (iii) 0.40; (iv) 0.45

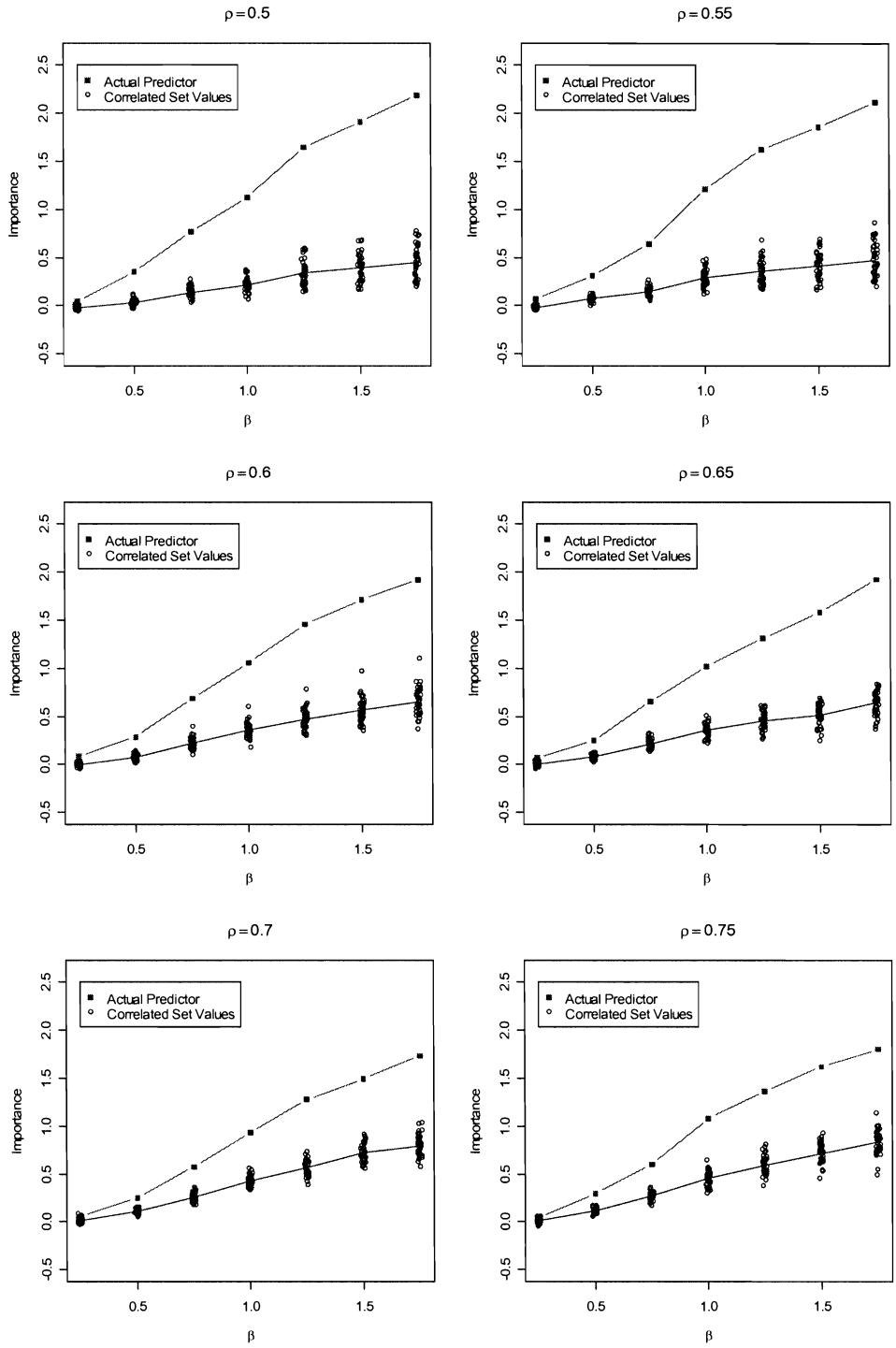

Figure 3c: Plot of the average variable importance for the actual predictor and the variables in the correlated set at all values of β when ρ = (i) 0.50; (ii) 0.55; (iii) 0.60; (iv) 0.65; (v) 0.70; (vi) 0.75

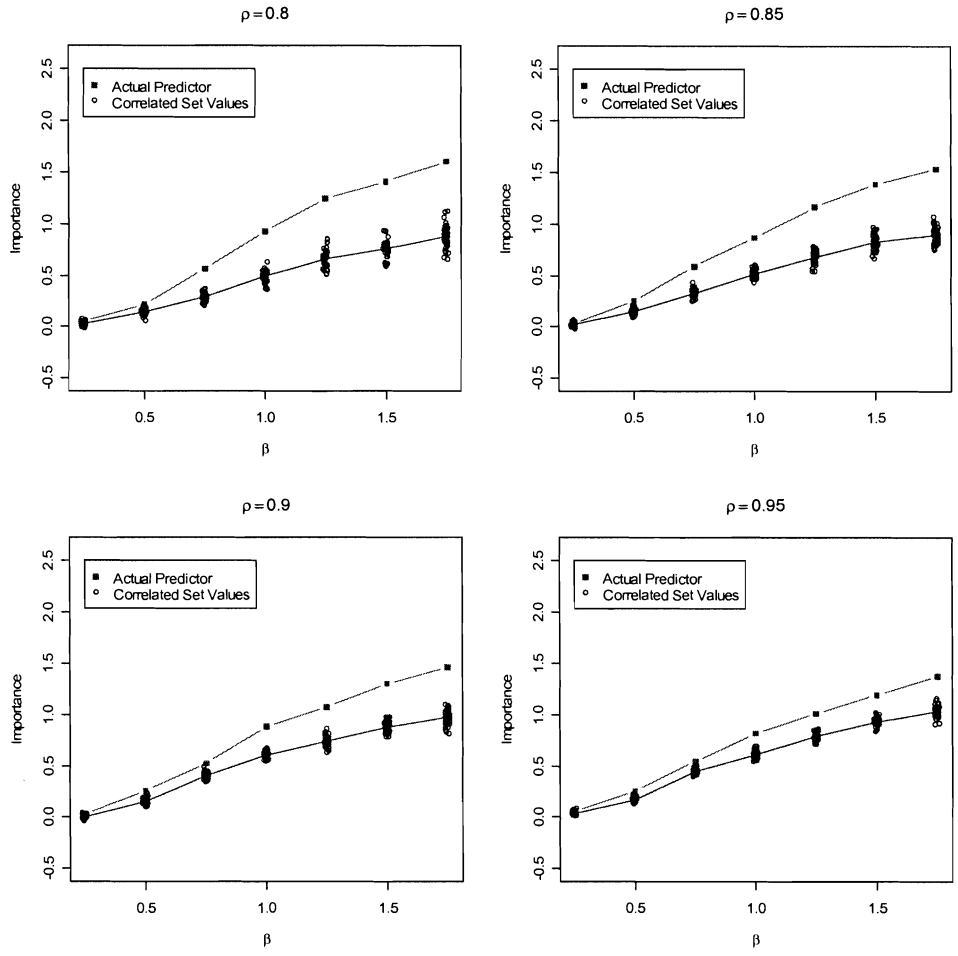

Figure 3d: Plot of the average variable importance for the actual predictor and the variables in the correlated set at all values of β when ρ = (i) 0.80; (ii) 0.85; (iii) 0.90; (iv) 0.95

As the set correlation increased, the variable importance for the variables in the set increased while the importance scores of the actual predictor decreased. Thus the variable importance of the actual predictor and those variables within its correlated set tend to coverge as the correlation increases. However, the actual predictor maintains an identifiably larger variable importance estimate with larger values of β. The variance of the importance estimates in the correlation sets increased as the correlation increased until the correlation was about 0.50, then the variance continued to decrease until the variance of the last correlation set is similar to that of the first set. In the 140 simulations, with different values of β and ρ, the importance score of the actual predictor was the maximum in all but 7 of these simulations. All 7 of those simulations had $\beta = 0.25$. The Gini variable importance estimates results were very similar to the mean decrease in accuracy importance score results.

The out-of-bag error for all forests was high (see Table 1). As expected, as β increased the oob error decreased. As the set correlation increased, the oob error decreases. The oob error rates are greater than 40% for all simulations where $\beta \leq 1.00$ and/or the set correlation is less than 0.35. When $\beta = 1.25$ and $\beta = 1.50$, the oob error rates drop below 40% when the set correlation approaches 0.50. When $\beta = 1.75$ the oob error rates drop below 40% when the set correlation is 0.35 and drop below 30% when the set correlation is 0.70. The best oob error rate is 26%, this is when $\beta = 1.75$ and the set correlation is 0.95. The worst oob error rate is 52.1% when $\beta = 0.25$ and the set correlation is 0.00.

Table 1: Out-of Bag Error (Averaged over $R = 100$ random forests)

	Beta						
Set Correlation	0.25	0.50	0.75	1.00	1.25	1.50	1.75
0.00	0.512	0.503	0.492	0.482	0.480	0.472	0.455
0.05	0.494	0.491	0.484	0.462	0.463	0.436	0.422
0.10	0.512	0.490	0.499	0.473	0.456	0.431	0.421
0.15	0.495	0.497	0.505	0.492	0.485	0.467	0.458
0.20	0.491	0.492	0.499	0.470	0.465	0.447	0.436
0.25	0.495	0.504	0.488	0.470	0.469	0.461	0.438
0.30	0.497	0.495	0.486	0.473	0.462	0.444	0.423
0.35	0.507	0.501	0.477	0.451	0.430	0.418	**0.384**
0.40	0.499	0.485	0.473	0.453	0.424	0.404	**0.389**
0.45	0.495	0.485	0.461	0.445	0.413	**0.386**	**0.357**
0.50	0.494	0.487	0.458	0.433	**0.388**	**0.366**	**0.344**
0.55	0.490	0.474	0.454	0.407	**0.393**	**0.363**	**0.351**
0.60	0.489	0.481	0.442	0.401	**0.367**	**0.345**	**0.314**
0.65	0.503	0.482	0.442	0.402	**0.381**	**0.360**	**0.325**
0.70	0.496	0.464	0.436	**0.388**	**0.361**	**0.314**	**0.293**
0.75	0.502	0.472	0.432	**0.386**	**0.345**	**0.315**	**0.290**
0.80	0.490	0.466	0.424	**0.374**	**0.332**	**0.303**	**0.275**
0.85	0.498	0.460	0.413	**0.384**	**0.344**	**0.313**	**0.297**
0.90	0.492	0.462	0.405	**0.369**	**0.334**	**0.305**	**0.281**
0.95	0.496	0.462	0.401	**0.360**	**0.329**	**0.289**	**0.260**

In the logistic regression simulation we found that the actual predictor was selected as the most important variable only 35.1% of the time. When all 40 variables from the correlated set were included in the model the algorithm failed to converge on 8 out of 10 attempts.

2.3 Simulation Study Conclusion

The simulation results were consistent with what was expected. As β increased, the relationship between the actual predictor and the response is stronger and therefore the variable importance estimates for the actual predictor increased. When the correlation between the actual predictor and the independent variables in its set increases, as expected, the importance estimates for those variables within the correlated set also increased. We also expected the importance estimates of the actual predictor to decrease as the correlation increases, since variables that are highly correlated with the actual predictor will do just as well at prediction as the actual predictor, thus lowering the importance of the actual predictor.

The high oob error rates were not unusual since only m variables are randomly selected at each node for splitting and the data were artificially generated to have only one actual predictor. The oob error rate decreased as β increased because the association between the response and the predictor was stronger. The oob error rate decreased as set correlation increased because, as previously mentioned, the variables that are highly correlated with the actual predictor became good predictors. When the actual predictor is missing from a tree, there is a higher chance that one of the variables that is highly correlated with it will get selected and do a good job of prediction.

Using a modeling algorithm such as random forests is more effective in this type of data analysis situation than traditional modeling techniques such as logistic regression. With only 40 highly correlated variables, forward logistic regression could not

consistently identify the actual predictor and if all variables in the correlated set were included in the logistic model, the algorithm usually did not converge on parameter estimates. In many applications, such as in gene expression analysis, there will be thousands of variables, many of which will be highly correlated. This is where traditional modeling techniques, such as logistic regression, break down, and algorithmic models such as random forests may excel.

CHAPTER 3 Random Forest Application to Microarray Data

3.1 Random Forest Application Introduction

Recent advances in microarray research have led to the development of an entirely new set of tools for statistical analysis. The size and scope of these gene expression data sets make them difficult to analyze using standard methods. A microarray chip can measure the presence of thousands of genes in one tissue sample. Typically a small sample of microarray chips will be collected to study classification, such as the difference in gene expression between cancerous and non-cancerous cells. The number of independent variables (genes) far exceeds the number of observations. Many of these variables will be highly correlated, so analytical methods must work in the presence of multicollinearity. Algorithmic modeling presents a viable solution to this difficult situation. We have shown that random forests is an effective tool for identifying a single predictor among a large number of independent variables. Now we will examine an application of random forests to a real microarray data set.

Hepatocellular carcinoma (HCC), or liver cancer, is one of the most common and aggressive human cancers. Liver cancer most commonly occurs in Asia and Africa, but its prevalence is increasing in Europe and North America. The incidence of HCC is increasing because of the spread of hepatitis B (HBV) and hepatitis C (HCV) viruses. HBV and HCV are directly responsible for carcinogenesis in the majority of cases.

The diagnosis for HCC is difficult and is usually diagnosed too late. Only a moderate improvement in long-term survival can be achieved through surgical

intervention and HCC has a high rate of recurrence after surgery. Despite recent surgical advances, about half of the patients with HCC die after hepatectomy. Research has been devoted to identifying the specific gene alterations associated with the disease but no predictive system has been developed to classify the morphology. Liver transplantation can be a curative treatment for small HCC but patients with potentially curable higher stage HCC are denied liver transplants because there is a lack of predictive progression and recurrence.

Data was collected on 38 patients with HCV-HCC, all candidates for liver transplantation. Gene expression analysis was performed on tumor samples using Affymetrix high-density oligonucleotide gene chips. During the study, 20 patients underwent liver transplant, 13 had cancer progression, 4 were still waiting for transplant, and 1 patient died without progression. Of interest is the ability to identify a suite of genes predictive of progression. In identifying these genes, the variable importance estimates from the random forest algorithms were successively examined to identify molecular markers associated with disease progression.

3.2 Random Forest Application Methods

Image analysis was performed using the Affymetrix software. For each probe, the perimeter pixel values are excluded and the probe intensity is calculated as the 75^{th} percentile of the interior pixels. Quality assessment was carried out by evaluating the 3':5' ratios for the GAPDH, B-actin, and ISGF probesets. After quality assessment

examinations, all control probe sets were removed and not considered in downstream analyses.

The Robust MultiArray Average method (RMA) was used for background subtraction, normalization, and expression summaries. First, the PM intensities were background adjusted on a raw intensity scale. Irizarry et al. (2003) motivated a background plus signal model: $PM_{ijn} = s_{ijn} + bg_{ijn}, i = 1,...,I, j = 1,...,J, n = 1,...,N$,

where s_{ijn} is the true signal and bg_{ijn} is the background signal for array i, probe j, and probe set n. The background corrected intensities, $PM^{*}_{ijn} = PM_{ijn} - \widehat{bg}_{ijn}$, are estimated by fitting a model assuming s_{ijn} follows an exponential distribution and bg_{ijn} follows a normal distribution.

Thereafter, quantile normalization was used to normalize the data, which assumes that the same underlying distribution is represented across all chips in an experiment. The motivation behind the quantile normalization method is the quantile-quantile plot, which observes a straight diagonal line if the distributions of two data vectors are the same. To enforce a set of vectors to have the same distribution in m dimensions, the points of the m dimensional quantile plot must project onto an m dimensional diagonal. (Bolstad et al. 2003). Therefore the quantile normalization algorithm is as follows:

1) Create a matrix X with i columns by combining the vectors of PM intensities for each chip.
2) Sort each column of X to form X_s.

3) Take the means across rows of matrix X_s. Assign this mean to each element in the row to get X_s^*.
4) To get the normalized intensity values X_{norm}, rearrange each column of X_s^* to have the same ordering as the original matrix X.

Now take the $\log_2$ of the normalized background corrected PM values to get the intensity values $PM_{ijn}^{**} = \log_2(qnormalized(PM_{ijn}^{*}))$.

The expression summaries are calculated by finding the RMA of these PM_{ijn}^{**}. This is accomplished by fitting the additive model

$$PM_{ijn}^{**} = \mu_{in} + \alpha_{jn} + \varepsilon_{ijn}, i = 1,...,I, j = 1,...,J, n = 1,...,N$$

where μ_i represents the log scale expression level for chip i and α_j is the probe affinity effect. We assume $\sum_{j=1}^{J} \alpha_j = 0$ for all probe sets. This assumes that the probe intensities are on average representative of the associated gene's expression. The model parameters are estimated using a median polish, which is a robust method similar to ANOVA. The median polish proceeds by initializing the column (chip) and row (probe affinity) effects to 0. Iteratively, row and column medians are estimated then subtracted from the data. The polished data is then used to estimate chip and probe affinity effects. Often only two iterations of row then column polishing are used to estimate the chip and probe affinity effects. The estimate of μ_{in} is the expression measure on chip i for probe set n.

After probe set expression summaries were obtained, random forests were applied to the microarray data. The number of trees grown was $B = 2500$, based on Breiman's

(2004) recommendation. The value for m, the number of predictor variables to choose from at each node, was left at default $(m = \sqrt{p})$. When there are a large number of noise variables, Liaw and Weiner (2002) suggest that increasing the value of m may give better performance. The value of m was not modified, however, since the out-of-bag error actually increased when m increased.

Inspired by methods suggested by Li et al. (2005) and Diaz Uriarte and Alvarez de Andres (2006), backwards elimination was used to narrow down the set of possible independent variables. A random forest was grown using the entire set of genes, $\boldsymbol{S}_{\text{all}}$, which included p = 22,215 probe sets. The resulting variable importance estimates were used to reduce the dimensionality of the dataset, by eliminating variables that have Gini importance measures less than $\overline{X} + \frac{3\sigma}{\sqrt{p}}$ to form a smaller subset of genes, $\boldsymbol{S}_1$. Here $\overline{X}$ represents the mean variable importance over the 22,215 probe sets in $\boldsymbol{S}_{\text{all}}$. The Gini importance measures were used because research suggests that the mean decrease in accuracy importance measures are not sensitive enough for applications with microarray data (Breiman 2004). If the predictive power of the individual variables is expected to be small and there are a large number of variables with a small number of observations, the Gini importance measures are more appropriate. This can be attributed to size of the out-of-bag sample $\mathcal{L}_{\text{oob}}$ used to calculate mean decrease in accuracy importance. Since the number of out-of-bag observations is approximately one-third of the total number of observations (n) in the learning sample $\mathcal{L}$, for small n, the importance measures will be granular. Gini importance measures are more appropriate for smaller sample sizes.

$\boldsymbol{S}_1$ was then used in growing a subsequent random forest. Again, unimportant variables were eliminated using the cutoff $\overline{X} + \frac{3\sigma}{\sqrt{p}}$. This process of eliminating unimportant variables was repeated until the set of predictor variables $\boldsymbol{S}_i$ is a manageable size ($n \approx 100$) and the out-of-bag error rate stabilized. The final set of probe sets is denoted $\boldsymbol{S}^*$.

Once a small set of important variables was identified, forward variable selection was employed. First, the variable importance estimates for probe sets in $\boldsymbol{S}^*$ were sorted in descending order, providing a list of variables $g_1,\ldots,g_p$, where g_j is the variable (probe set) with importance rank j. Then for step *1*, a random forest using the variables g_1 and g_2 was derived and the out-of-bag error rate estimated. At the $k = 2$ step, a random forest was derived after adding g_3 to the previous forest and again the error rate was estimated. For $k=1,\ldots,p$, this procedure was repeated, adding g_{k+1} at each step. The random forest model with the lowest out-of-bag error rate was chosen as the most parsimonious model. This parsimonious set of important variables is denoted $\boldsymbol{S}_{\text{final}}$.

The gene ontology (GO) database was used to lookup genes present in $\boldsymbol{S}^*$ and in the full sample $\boldsymbol{S}_{\text{all}}$ and match those genes with designated molecular functions, biological processes, and cellular components (GO terms) determined by the GO database. The GO library in R features functions to compare the proportion of genes matching each GO term that are present in $\boldsymbol{S}^*$ to the proportion in $\boldsymbol{S}_{\text{all}}$. Note that the proportions in each gene set will not necessarily sum to one because the GO database may categorize a gene as being associated with more than one function or process. If the genes involved in a specific function are more common in $\boldsymbol{S}^*$ this indicates that this function is most related

to the outcome. This may suggest further research into molecular functions and processes that may predict progression, rather than limiting the focus to specific genes that have been identified as important. The GO database and the Gene Ontology Annotation (GOA) project are still under development. The specific genes associated with various functions and processes are continually updated and can be inaccurate. This type of analysis should be considered exploratory and conclusions should be considered tentative.

3.3 Random Forest Application Results

There was no evidence for lack of quality in the array sets. All 3':5' ratios were below 3, the tolerable threshold recommended by Affymetrix. Four rounds of elimination were used to narrow down the number of predictor variables to form $\boldsymbol{S}^*$ ($n_{final} = 102$). The names of the genes in the variable set $\boldsymbol{S}^*$ are presented in order of importance in Appendix C. Figures 4a-b illustrate the importance measures of the predictors at each stage of elimination. For each of the backward elimination steps, the variable importance estimated for the genes included in the final set are designated by a different plotting symbol.

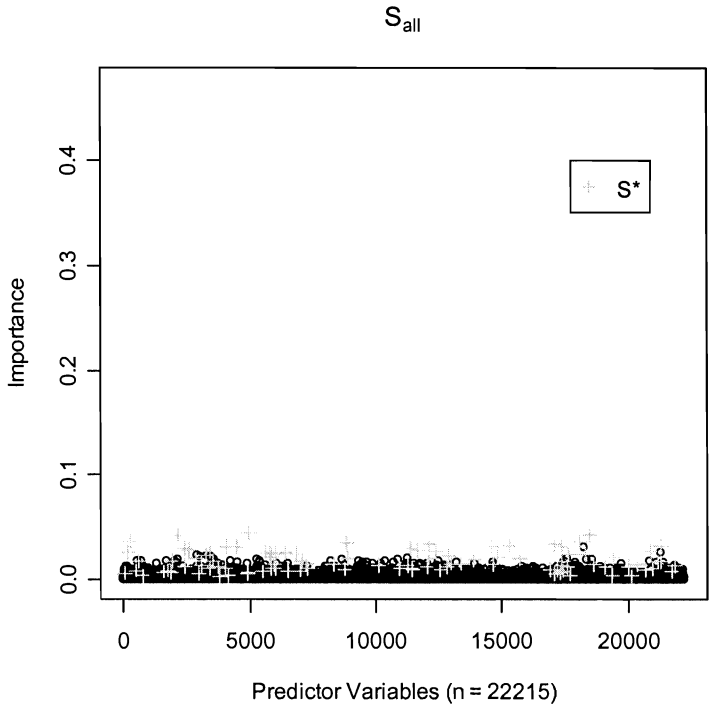

Figure 4a: Gini Importance Estimates for each probe set when all probe sets (n=22,215) were included in the random forest. Probe sets in the final set S* are plotted using (+) whereas all other probe sets are designated with (o).

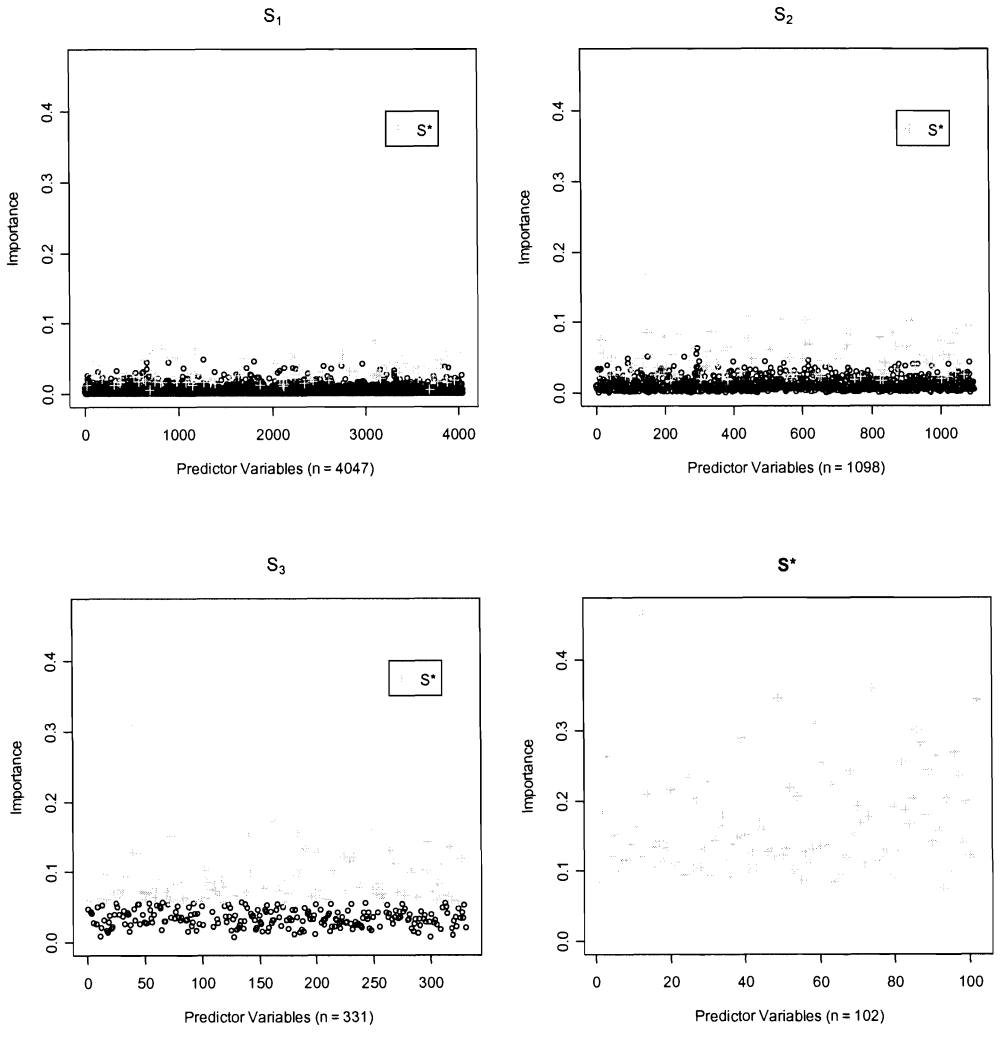

Figure 4b: Gini Importance Estimates for each probe set after backwards elimination steps (i) S_1; (ii) S_2; (iii) S_3; (iv) S*;. Probe sets in the final set S* are plotted using (+) whereas all other probe sets are designated with (o).

The mean plus three standard error cutoff values ($\overline{X} + \frac{3\sigma}{\sqrt{p}}$) assume that the Gini importance scores are normally distributed. The importance scores appeared to be exponentially distributed, but when the equivalent cutoff values assuming an exponential distribution ($\overline{X} + \frac{3\overline{X}}{\sqrt{p}}$) were calculated, the values were almost exactly the same due to

such a large sample size. Most of the variables in S^* have the highest importance scores in all the random forests. This substantiates Liaw and Weiner's (2002) claim that the ranking of variable importance measures is quite stable.

The out-of-bag error rates of the random forests decreased as variables were eliminated. This can be explained by the enormous complexity of the classification trees in the forests when thousands of variables are being used for prediction. As the number of variables decreases the noise variables are eliminated and the predictions stabilize. Forward selection proceeding with probe sets with the largest variable importance estimates suggested a final set of four probe sets, denoted S_{final}, having the lowest out-of-bag error rate, see Table 2.

Table 2: Forest Out-of-Bag Error (Overall and Class-wise) including each probe set after backwards elimination and the final probe set from forward selection

	Number of Probe Sets	Overall	No Progression	Progression
S_{all}	22215	0.376	0.122	0.862
S_1	4047	0.345	0.098	0.82
S_2	1098	0.216	0.048	0.539
S_3	331	0.188	0.046	0.462
S^*	102	0.187	0.045	0.461
S_{final}	**4**	**0.057**	**0.038**	**0.094**

The out-of-bag class-specific error rate for patients without progression was 3.8% while the error rate for patients with progression was 9.4%, leading to an overall out-of-bag error of 5.7% for S_{final}. Details pertaining to the four probe sets can be found in Table 3.

Table 3: Information on the Four Probe Sets in S_{final}

Gene ID	Affymetrix ID	Symbol	UniGene Cluster	Name	Chromose Band
54541	202887_s_at	DDIT4	Hs.523012	DNA-damage-inducible transcript 4	10pter-q26.12
10135	217738_at	PBEF1	Hs.489615	pre-B-cell colony enhancing factor 1	7q22.3
10111	209349_at	RAD50	Hs.242635	RAD50 homolog (S. cerevisiae)	5q31
6453	35776_at	ITSN1	Hs.160324	intersectin 1 (SH3 domain protein)	21q22.1-q22.2

The gene ontology of the set of genes S^* is compared to the full set of genes S_{all}. Figures 5 and 6 illustrate a comparison of the proportions of genes present for various molecular functions. The bars represent the proportion of genes in S^* and S_{all} that have been matched to particular molecular functions. Figure 5 features the proportions for all the GO molecular function terms. Figure 6 features only the GO molecular function terms where the proportions in S^* exceed those in S_{all}.

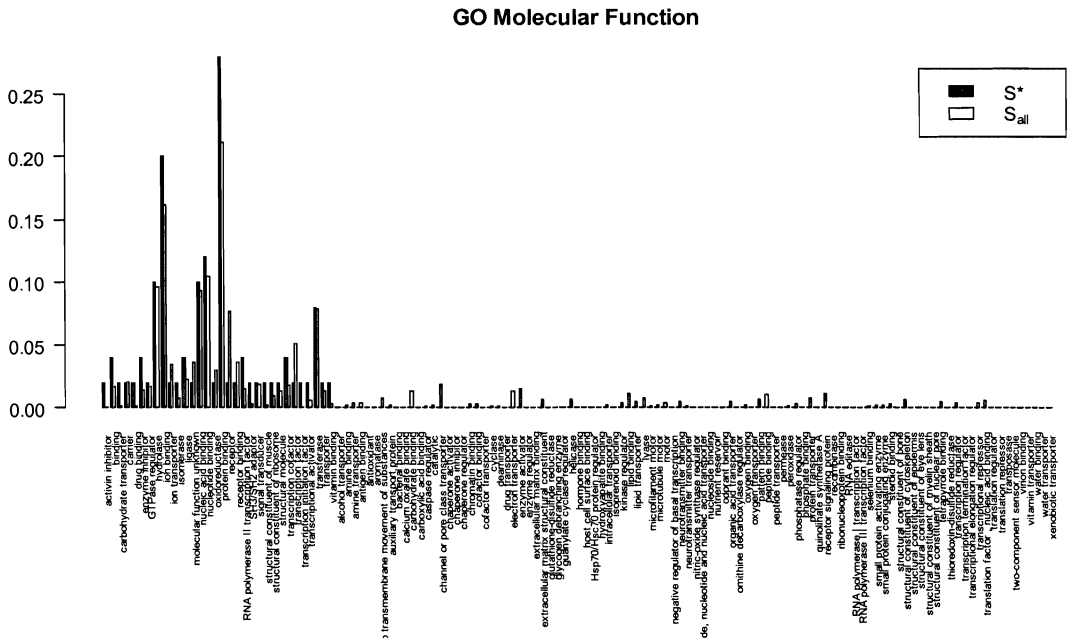

Figure 5: GO Molecular Function Proportions for the probe sets in S^* and in the entire gene chip S_{all}

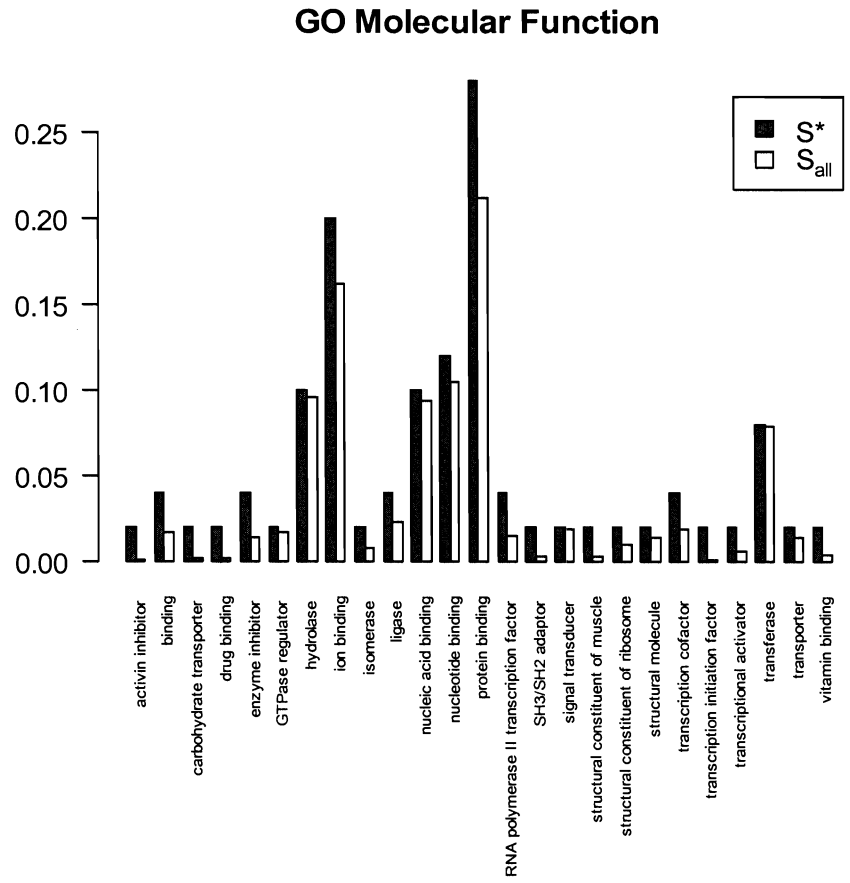

Figure 6: GO Molecular Function Proportions for the probe set S* and in the entire gene chip S_{all} with S^*:S_{all} Ratios Greater than 1

The ratio of the proportion of genes present in S^* is compared to S_{all} for each molecular function. A molecular function is important for predicting progression if $\frac{S^*}{S_{all}} > 1$. There are 25 molecular functions that are more present in S^*. In 13 of those molecular functions the ratio is greater than 2. Genes involved in activin inhibitor activity and transcription initiation factor activity are much more present in S^* than S_{all}, with

ratios as high as 88.9 and 34.2, respectively. Table 4 lists the molecular functions and their ratios.

Table 4: Molecular Functions with S^*:S_{alll} Ratios Greater than 1

Molecular Function	Ratio(S^*:S_{alll})
activin inhibitor activity	**88.860**
transcription initiation factor activity	**34.177**
drug binding	**12.008**
carbohydrate transporter activity	**11.692**
structural constituent of muscle	**7.660**
SH3/SH2 adaptor activity	**6.534**
vitamin binding	**6.171**
transcriptional activator activity	**3.316**
enzyme inhibitor activity	**2.866**
RNA polymerase II transcription factor activity	**2.653**
isomerase activity	**2.629**
binding	**2.351**
transcription cofactor activity	**2.183**
structural constituent of ribosome	1.983
ligase activity	1.781
structural molecule activity	1.506
transporter activity	1.466
protein binding	1.326
ion binding	1.235
GTPase regulator activity	1.169
nucleotide binding	1.152
signal transducer activity	1.084
nucleic acid binding	1.073
hydrolase activity	1.044
transferase activity	1.012

Figure 7 illustrates a comparison of the proportions of genes present that are involved in different biological processes and cellular components.

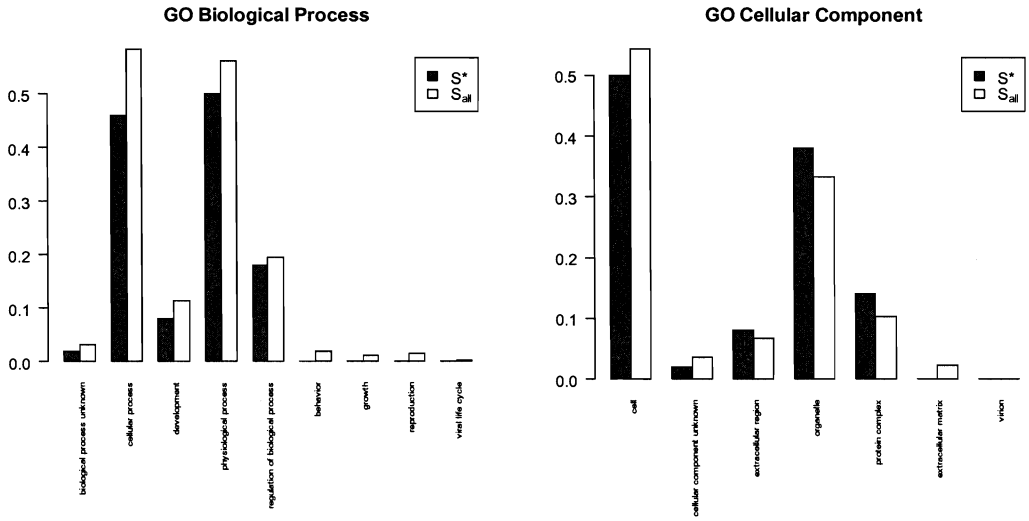

Figure 7: GO Biological Process and GO Cellular Component Proportions for the probe sets in S* and in the entire gene chip S_{all}

There are no biological processes where the genes are more present in S^*. There are three cellular components with ratios greater than 1, but all are very close to 1.

Table 5: Cellular Components with S^*:S_{all} Ratios Greater than 1

Cellular Component	Ratio(S^*/S_{all})
protein complex	1.358
extra cellular region	1.187
organelle	1.142

3.4 Random Forest Application Conclusion

The use of random forests on this HCV-HCC microarray data helped us gather useful information for predicting progression. Identifying four genes that can predict progression with such a low out-of-bag error rate (5.7%) should prove to be clinically relevant and useful. Such a small set of genes is advantageous because it facilitates further research using more focused techniques such as clinical assays. Clinical assays can be used to interrogate a small subset of genes that have strong predictive power. This can lead to simpler, less invasive testing techniques to predict progression. Microarray experiments are costly, complicated, and the tissue samples are usually obtained through biopsies or other invasive procedures. If progression can be predicted by testing for genes in a urine sample or blood test, this will save the patient a lot of pain and simplify the diagnosis process for the clinician. The larger set of variables $\boldsymbol{S}^*$ may also be helpful in providing an area of focus for further research. There was no strong evidence of biological processes or cellular components that predict disease progression, but 13 molecular functions were proportionally at least two times over-represented in $\boldsymbol{S}^*$ than in the full set of genes on the GeneChip. Genes involved in inhibitor activity and transcription initiation factor activity had a very strong presence in $\boldsymbol{S}^*$ and may play a vital role in tumor progression. We have been able to show that random forests can identify important variables in gene expression analysis where traditional modeling methods are unreliable.

CHAPTER 4 Future Work

Simulations were used to study the effectiveness of random forest importance measures at identifying one actual predictor among many independent variables. Most of the time there will be a number of predictors associated with a response. Additional simulations could be performed to study common models in which there is a complex association between the predictors and the response. The random forest algorithm was implemented using default settings based on suggestions in the literature (Breiman 2001a; Svetnik 2003; Diaz-Uriarte and Alvarez de Andres 2006). The value of m, the number of independent variables sampled at each node, is considered one of the only adjustable parameters in random forests. It is noted, however, that the variable importance estimates and out-of-bag error rates may have been different for different values of m.

Breiman (2004) suggested four methods for estimating variable importance measures using the random forest framework. We have studied the mean decrease in accuracy and the Gini importance measures and found them to give similar results. These are the only importance measures available for the randomForest library in R. Other measures of importance should be studied and could be added to this R package. A modification to the mean decrease in accuracy importance measure is suggested. Using the standard mean decrease in accuracy measure, if the actual predictor is not selected to be in a tree, permuting the out-of-bag values for the actual predictor will have no effect on the prediction accuracy. The actual predictor will not show up as important. Other

variables that are highly correlated with the predictor are masking its role. Instead of permuting the values of the out-of-bag observations one column at a time, cluster the independent variables and permute the row values of the entire cluster.

Developments in the algorithmic modeling community have led to a number of competitors for random forests. Boosting is a modified version of bagging. Instead of using a simple bootstrap sample of the learning set to grow each tree, the algorithm iteratively weights observations in the learning set so that observations that were incorrectly predicted by previous classifiers have a higher probability of being selected in the next sample. Therefore boosting attempts to improve the set of classification trees by producing new trees that are better able to predict observations for which the current set is doing a poor job. Boosting can greatly outperform bagging although there are cases where boosting can do worse than a single tree classifier. Opitz and Maclin (1999) found that bagging is a stable procedure which is a good choice for most problems, but when appropriate, boosting can produce large gains in accuracy.

Dettling and Buhlman (2003) used 6 real and simulated data sets to demonstrate the effectiveness of boosting for predicting gene expression data. Another study by Dettling (2004) developed a method using a mixture of bagging and boosting called BagBoosting. In the boosting algorithm reweighting is applied after each tree is grown. BagBoosting grows a set of classification trees, calculates the error, and reweights based on the bagged tree misclassification. A new set of bagged trees is grown to better predict observations that were misclassified in the last set of bagged trees. The classifier is thus a set of bagged tree sets. Real and simulated gene expression data were used to show that

BagBoosting can compete with random forests and has consistently lower misclassification error compared with bagging and boosting.

Hothorn (2005) describes a method of bundling classifiers. Bagged classification trees are grown and the out-of-bag observations are used to grow additional classifiers, such as linear discriminant analysis and k-nearest neighbors. All of the classifiers are combined to vote for predictions (Hothorn and Lausen 2003; Hothorn 2005).

It is not yet clear as to whether random forests will emerge as a practical tool for gene expression analysis. Even with robust methods such as random forests, methods for estimating variable importance and methods of variable elimination are crucial areas of research that must be addressed. It remains to be seen whether modifications to current algorithms provide better variable importance measures. Variable selection in these microarray studies can be very difficult and proper gene selection is critical in making this type of research useful. Nevertheless, algorithmic models such as bagging and random forests appear to have a promising future. More research is needed to study the effectiveness of these algorithms, further refine them, and develop new ones.

References

References

Bolstad, B. M., Irizarray, R. A., Astrand, M., and Speed, T. P. (2003), "A Comparison of Normalization Methods for High Density Oligonucleotide Array Data Based on Variance and Bias," *Bioinformatics*, 19, 185-193.

Breiman, L. (1996), "Bagging Predictors," *Machine Learning*, 24, 123-140.

Breiman, L., Friedman, J. H., Olshen, R. A., and Stone, C. J. (1998), *Classification and Regression Trees*, Boca Raton, FL: Chapman & Hall/CRC.

Breiman, L. (2001a), "Random Forests," *Machine Learning*, 45, 5-32.

Breiman, L. (2001b), "Statistical Modeling: The Two Cultures," *Statistical Science*, 16, 199-231.

Breiman, L. (2004), "Manual on Setting up, Using and Understanding Random Forests V3.1," http://www.stat.berkeley.edu/users/breiman/RandomForests/cc_manual.htm, June 2004.

Breiman, L., and Cutler, A. (2005), "Random Forests – Classification Description.", http://www.stat.berkeley.edu/users/breiman/RandomForests/cc_home.htm, October 2005.

Briem, G. J., Benediktsson, J. A., and Sveinsson, J. R. (2001), " Boosting, Bagging, and Consensus Based Classification of Multisource Remote Sensing Data," in *Multiple Classifier Systems*, eds. J. Kittler and F. Roli, Berlin Heidelberg: Springer-Verlag, pp. 279-288.

Bryll, R., Gutierrez-Osuna, R., and Quek, F. (2003), "Attribute Bagging: Improving Accuracy of Classifier Ensembles by Using Random Feature Subsets," *Pattern Recognition*, 35, 1291-1302.

Bühlmann, P. (2004), "Bagging, Boosting and Ensemble Methods," in *Handbook of Computational Statistics*, Berlin: Springer, pp. 877-907.

Dettling, M., and Buhlmann, P. (2003), "Boosting for Tumor Classification with Gene Expression Data," *Bioinformatics*, 19, 1061-1069.

Dettling, M. (2004), "Bagboosting for Tumor Classification with Gene Expression Data," *Bioinformatics*, 20, 3583-3593.

Diaz-Uriarte, R., and Andres, S. A. d. (2006), "Gene Selection and Classification of Microarray Data Using Random Forest," *BMC Bioinformatics*, 7, 1471-2105.

Dietterich, T. G. (2003), "Ensemble Learning," in *The Handbook of Brain Theory and Neural Networks*, ed. M. A. Arbib, Cambridge, MA: MIT Press.

Gentleman, R., Huber, W., Carey, V. J., Irizarry, R. A., and Dudoit, S. (2005), *Bioinformatics and Computational Biology Solutions Using R and Bioconductor*, New York, NY: Springer.

Hastie, T., Tibshirani, R., and Friedman, J. (2001), *The Elements of Statistical Learning*, New York, NY: Springer.

Hothorn, T., and Lausen, B. (2003), "Bagging Tree Classifiers for Laser Scanning Images: A Data- and Simulation-Based Strategy," *Artificial Intelligence in Medicine*, 27, 65-79.

Hothorn, T., and Lausen, B. (2005), "Bundling Classifiers by Bagging Trees," *Computational Statistics and Data Analysis*, 49, 1068-1078.

Iizuka, N., Hamamoto, Y., and Oka, M. (2004), "Predicting Individual Outcomes in Hepatocellular Carcinoma," *The Lancet*, 364.

Irizarry, R. A., Hobbs, B., and Collin, F. (2003), "Exploration, Normalization, and Summaries of High Density Oligonucleotide Array Probe Level Data," *Biostatistics*, 4, 249-264.

Izmirlian, G. (2004), "Application of the Random Forest Classification Algorithm to a Seldi-Tof Proteomics Study in the Settingn of a Cancer Prevention Trial," *Annals New York Academy of Sciences*, 1020, 154-174.

Li, S., Fedorowicz, A., Singh, H., and Soderholm, S. C. (2005), "Application of the Random Forest Method in Studies of Local Lymph Node Assay Based Skin Sensitization Data," *Journal of Chemical Information and Computer Sciences*, 45, 952-964.

Liaw, A., and Wiener, M. (2002), "Classification and Regression by Random Forest," *R News*, 2/3.

Marsh, J. W., Finkelstein, S. D., and Demetris, A. J. (2003), "Genotyping of Hepatocellular Carcinoma in Liver Transplant Recipients Adds Predictive Power for Determining Recurrence-Free Survival," *Liver Transplantation*, 9, 664-671.

Nock, R. (2002), "Inducing Interpretable Voting Classifiers without Trading Accuracy for Simplicity: Theoretical Results, Approximation Algorithms, and Experiments," *Journal of Artificial Intelligence Research*, 17, 137-170.

Olthoff, K. M., Merion, R. M., Ghobrial, R., Abecassis, M. M., and Everhart, J. E. (2005), "Outcomes of 385 Adult-to-Adult Living Donor Liver Transplant Recipients," *Annals of Surgery*, 242, 314-325.

Opitz, D., and Maclin, R. (1999), "Popular Ensemble Methods: An Empirical Study," *Journal of Artificial Intelligence Research*, 11, 169-198.

Pavlidis, P., Qin, J., Arango, V., Mann, J. J., and Sibille, E. (2004), "Using the Gene Ontology for Microarray Data Mining: A Comparison of Methods and Application to Age Effects in Human Prefrontal Cortex," *Neurochemical Research*, 29, 1213-1222.

Peters, J., Samson, R., and Verhoest, N. E. C. (2005), "Predictive Ecohydrological Modelling Using the Random Forest Algorithm," *Comm. Appl. Biol. Sci*, 70, 207-211.

R Development Core Team (2005), "R: A Language and Environment for Statistical Computing," *R Foundation for Statistical Computing*.

Ripley, B. D. (1996), "Tree-Structured Classifiers," in *Pattern Recognition and Neural Networks*, Cambridge University Press.

Skurichina, M., and Duin, R. P. W. (2001), "Bagging and the Random Subspace Method for Redundant Feature Spaces," in *Multiple Classifier Systems*, eds. J. Kittler and F. Roli, Berlin Heidelberg: Springer-Verlag, pp. 1-10.

Steinberg, D., and Colla, P. (1997), *Cart - Classification and Regression Trees*, San Diego, CA: Salford Systems.

Svetnik, V., et al. (2003), "Random Forest: A Classification and Regression Tool for Compound Classification and Qsar Modeling," *J. Chem. Inf. Comput. Sci.*, 43, 1947-1958.

APPENDIX A - Simulation Summary Tables

Table I: Summary statistics of mean decrease in accuracy variable importance measures for all simulations (Averaged over $R = 100$ random forests)

Mean Decrease in Accuracy Variable Importance							
simulation	ρ	β	Overall Mean	Overall Std.	Correlated Set Mean	Correlated Set Std.	Actual Predictor
1	0.00	0.25	-0.032	0.020	-0.035	0.021	0.036
2	0.00	0.50	-0.032	0.027	-0.033	0.024	0.467
3	0.00	0.75	-0.030	0.035	-0.034	0.022	0.741
4	0.00	1.00	-0.030	0.054	-0.029	0.025	1.318
5	0.00	1.25	-0.032	0.069	-0.038	0.029	1.719
6	0.00	1.50	-0.034	0.083	-0.035	0.024	2.102
7	0.00	1.75	-0.030	0.089	-0.030	0.033	2.235
8	0.05	0.25	-0.029	0.022	-0.035	0.022	0.069
9	0.05	0.50	-0.027	0.027	-0.032	0.026	0.388
10	0.05	0.75	-0.028	0.037	-0.037	0.026	0.775
11	0.05	1.00	-0.024	0.059	-0.032	0.027	1.415
12	0.05	1.25	-0.025	0.070	-0.032	0.038	1.747
13	0.05	1.50	-0.021	0.083	-0.031	0.032	2.081
14	0.05	1.75	-0.020	0.094	-0.029	0.039	2.343
15	0.10	0.25	-0.033	0.021	-0.034	0.020	0.076
16	0.10	0.50	-0.028	0.027	-0.036	0.020	0.390
17	0.10	0.75	-0.031	0.039	-0.041	0.019	0.783
18	0.10	1.00	-0.026	0.056	-0.031	0.025	1.309
19	0.10	1.25	-0.023	0.074	-0.033	0.034	1.771
20	0.10	1.50	-0.018	0.085	-0.029	0.039	2.031
21	0.10	1.75	-0.017	0.096	-0.030	0.043	2.230
22	0.15	0.25	-0.031	0.021	-0.032	0.022	0.043
23	0.15	0.50	-0.030	0.026	-0.029	0.026	0.454
24	0.15	0.75	-0.031	0.036	-0.019	0.024	0.755
25	0.15	1.00	-0.031	0.052	-0.018	0.049	1.221
26	0.15	1.25	-0.032	0.069	-0.012	0.063	1.713
27	0.15	1.50	-0.031	0.084	0.007	0.080	2.083
28	0.15	1.75	-0.033	0.092	0.006	0.085	2.276

Mean Decrease in Accuracy Variable Importance							
simulation	ρ	β	Overall Mean	Overall Std.	Correlated Set Mean	Correlated Set Std.	Actual Predictor
29	0.20	0.25	-0.029	0.021	-0.032	0.018	0.077
30	0.20	0.50	-0.029	0.026	-0.021	0.022	0.399
31	0.20	0.75	-0.031	0.037	-0.029	0.032	0.784
32	0.20	1.00	-0.025	0.054	-0.019	0.033	1.289
33	0.20	1.25	-0.027	0.065	-0.008	0.045	1.598
34	0.20	1.50	-0.023	0.080	-0.003	0.058	1.971
35	0.20	1.75	-0.024	0.091	-0.002	0.067	2.238
36	0.25	0.25	-0.029	0.022	-0.031	0.023	0.117
37	0.25	0.50	-0.031	0.024	-0.029	0.027	0.324
38	0.25	0.75	-0.030	0.038	-0.024	0.029	0.836
39	0.25	1.00	-0.028	0.055	-0.011	0.035	1.345
40	0.25	1.25	-0.029	0.073	-0.006	0.056	1.820
41	0.25	1.50	-0.029	0.083	0.001	0.069	2.096
42	0.25	1.75	-0.027	0.093	0.006	0.069	2.347
43	0.30	0.25	-0.031	0.020	-0.027	0.018	0.032
44	0.30	0.50	-0.029	0.025	-0.010	0.026	0.318
45	0.30	0.75	-0.027	0.037	0.015	0.036	0.743
46	0.30	1.00	-0.027	0.054	0.023	0.046	1.265
47	0.30	1.25	-0.025	0.071	0.067	0.062	1.682
48	0.30	1.50	-0.024	0.081	0.079	0.076	1.925
49	0.30	1.75	-0.023	0.094	0.102	0.095	2.219
50	0.35	0.25	-0.032	0.021	-0.017	0.020	0.083
51	0.35	0.50	-0.029	0.025	0.001	0.026	0.334
52	0.35	0.75	-0.025	0.041	0.042	0.041	0.820
53	0.35	1.00	-0.021	0.059	0.102	0.075	1.200
54	0.35	1.25	-0.020	0.082	0.165	0.093	1.732
55	0.35	1.50	-0.019	0.094	0.171	0.119	2.022
56	0.35	1.75	-0.012	0.110	0.218	0.154	2.306
57	0.40	0.25	-0.031	0.020	-0.018	0.022	0.053
58	0.40	0.50	-0.027	0.029	0.023	0.027	0.475
59	0.40	0.75	-0.025	0.042	0.053	0.042	0.798
60	0.40	1.00	-0.021	0.063	0.116	0.068	1.272
61	0.40	1.25	-0.018	0.080	0.173	0.079	1.645
62	0.40	1.50	-0.012	0.095	0.214	0.102	1.962
63	0.40	1.75	-0.013	0.105	0.230	0.130	2.172
64	0.45	0.25	-0.029	0.021	-0.009	0.020	0.103
65	0.45	0.50	-0.026	0.026	0.028	0.031	0.296

Mean Decrease in Accuracy Variable Importance							
simulation	ρ	β	Overall Mean	Overall Std.	Correlated Set Mean	Correlated Set Std.	Actual Predictor
66	0.45	0.75	-0.020	0.048	0.118	0.056	0.737
67	0.45	1.00	-0.016	0.063	0.167	0.068	1.076
68	0.45	1.25	-0.013	0.087	0.256	0.098	1.425
69	0.45	1.50	-0.007	0.100	0.302	0.119	1.676
70	0.45	1.75	0.000	0.125	0.415	0.134	1.981
71	0.50	0.25	-0.030	0.021	-0.019	0.018	0.044
72	0.50	0.50	-0.027	0.029	0.032	0.032	0.352
73	0.50	0.75	-0.019	0.052	0.137	0.055	0.767
74	0.50	1.00	-0.015	0.072	0.213	0.072	1.125
75	0.50	1.25	-0.009	0.107	0.340	0.123	1.646
76	0.50	1.50	-0.005	0.122	0.395	0.139	1.913
77	0.50	1.75	-0.002	0.138	0.449	0.162	2.189
78	0.55	0.25	-0.029	0.021	-0.018	0.017	0.067
79	0.55	0.50	-0.024	0.032	0.071	0.029	0.308
80	0.55	0.75	-0.019	0.049	0.143	0.046	0.642
81	0.55	1.00	-0.012	0.086	0.286	0.088	1.212
82	0.55	1.25	-0.009	0.108	0.356	0.124	1.627
83	0.55	1.50	-0.002	0.123	0.412	0.144	1.861
84	0.55	1.75	-0.001	0.141	0.475	0.170	2.114
85	0.60	0.25	-0.029	0.021	-0.003	0.020	0.086
86	0.60	0.50	-0.025	0.033	0.070	0.032	0.286
87	0.60	0.75	-0.016	0.065	0.221	0.056	0.688
88	0.60	1.00	-0.008	0.096	0.356	0.074	1.059
89	0.60	1.25	-0.002	0.125	0.475	0.101	1.456
90	0.60	1.50	0.005	0.147	0.565	0.124	1.712
91	0.60	1.75	0.010	0.168	0.654	0.145	1.920
92	0.65	0.25	-0.031	0.021	0.003	0.023	0.066
93	0.65	0.50	-0.025	0.032	0.080	0.028	0.248
94	0.65	0.75	-0.016	0.062	0.214	0.048	0.658
95	0.65	1.00	-0.009	0.096	0.360	0.067	1.021
96	0.65	1.25	-0.005	0.119	0.456	0.090	1.315
97	0.65	1.50	0.000	0.136	0.516	0.107	1.583
98	0.65	1.75	0.006	0.165	0.641	0.121	1.924
99	0.70	0.25	-0.027	0.022	0.013	0.026	0.055
100	0.70	0.50	-0.021	0.037	0.104	0.023	0.252
101	0.70	0.75	-0.013	0.068	0.254	0.045	0.574
102	0.70	1.00	-0.002	0.106	0.427	0.056	0.935
103	0.70	1.25	0.005	0.138	0.563	0.081	1.276

Mean Decrease in Accuracy Variable Importance							
simulation	ρ	β	Overall Mean	Overall Std.	Correlated Set Mean	Correlated Set Std.	Actual Predictor
104	0.70	1.50	0.018	0.172	0.724	0.092	1.493
105	0.70	1.75	0.024	0.190	0.795	0.111	1.732
106	0.75	0.25	-0.030	0.023	0.012	0.025	0.039
107	0.75	0.50	-0.025	0.040	0.114	0.027	0.297
108	0.75	0.75	-0.014	0.072	0.269	0.042	0.601
109	0.75	1.00	-0.006	0.115	0.453	0.076	1.083
110	0.75	1.25	0.003	0.147	0.593	0.099	1.365
111	0.75	1.50	0.009	0.174	0.715	0.110	1.621
112	0.75	1.75	0.016	0.200	0.831	0.124	1.802
113	0.80	0.25	-0.026	0.024	0.027	0.022	0.054
114	0.80	0.50	-0.021	0.044	0.144	0.031	0.217
115	0.80	0.75	-0.012	0.075	0.290	0.036	0.565
116	0.80	1.00	0.001	0.118	0.487	0.057	0.926
117	0.80	1.25	0.010	0.158	0.665	0.082	1.242
118	0.80	1.50	0.018	0.179	0.760	0.085	1.411
119	0.80	1.75	0.024	0.205	0.876	0.107	1.602
120	0.85	0.25	-0.030	0.023	0.019	0.019	0.033
121	0.85	0.50	-0.019	0.045	0.151	0.026	0.252
122	0.85	0.75	-0.008	0.083	0.328	0.042	0.590
123	0.85	1.00	0.002	0.123	0.514	0.039	0.869
124	0.85	1.25	0.013	0.160	0.681	0.061	1.166
125	0.85	1.50	0.022	0.192	0.828	0.072	1.386
126	0.85	1.75	0.023	0.208	0.893	0.077	1.533
127	0.90	0.25	-0.029	0.022	0.006	0.016	0.028
128	0.90	0.50	-0.021	0.046	0.154	0.029	0.254
129	0.90	0.75	-0.006	0.098	0.405	0.030	0.522
130	0.90	1.00	0.006	0.141	0.600	0.038	0.884
131	0.90	1.25	0.014	0.172	0.741	0.053	1.070
132	0.90	1.50	0.025	0.202	0.878	0.054	1.301
133	0.90	1.75	0.026	0.223	0.969	0.071	1.463
134	0.95	0.25	-0.028	0.025	0.039	0.016	0.058
135	0.95	0.50	-0.018	0.049	0.171	0.024	0.255
136	0.95	0.75	-0.006	0.106	0.442	0.028	0.544
137	0.95	1.00	0.002	0.144	0.612	0.038	0.817
138	0.95	1.25	0.010	0.182	0.790	0.040	1.013
139	0.95	1.50	0.020	0.212	0.929	0.042	1.193
140	0.95	1.75	0.025	0.236	1.033	0.052	1.375

Table II: Summary statistics of Gini variable importance measures for all simulations (Averaged over $R = 100$ random forests)

Gini Variable Importance							
simulation	ρ	β	Overall Mean	Overall Std.	Correlated Set Mean	Correlated Set Std.	Actual Predictor
1	0.00	0.25	0.061	0.008	0.068	0.004	0.102
2	0.00	0.50	0.061	0.012	0.068	0.005	0.287
3	0.00	0.75	0.061	0.016	0.067	0.005	0.439
4	0.00	1.00	0.061	0.027	0.068	0.007	0.778
5	0.00	1.25	0.061	0.038	0.066	0.008	1.072
6	0.00	1.50	0.061	0.050	0.066	0.009	1.429
7	0.00	1.75	0.061	0.056	0.065	0.011	1.573
8	0.05	0.25	0.061	0.008	0.068	0.004	0.105
9	0.05	0.50	0.061	0.011	0.068	0.007	0.242
10	0.05	0.75	0.061	0.017	0.066	0.006	0.464
11	0.05	1.00	0.061	0.030	0.065	0.007	0.850
12	0.05	1.25	0.061	0.038	0.067	0.011	1.099
13	0.05	1.50	0.062	0.049	0.066	0.011	1.406
14	0.05	1.75	0.061	0.060	0.065	0.014	1.693
15	0.10	0.25	0.061	0.008	0.067	0.004	0.119
16	0.10	0.50	0.061	0.010	0.067	0.005	0.241
17	0.10	0.75	0.061	0.017	0.067	0.007	0.460
18	0.10	1.00	0.061	0.027	0.066	0.008	0.765
19	0.10	1.25	0.062	0.039	0.066	0.011	1.115
20	0.10	1.50	0.062	0.048	0.063	0.013	1.368
21	0.10	1.75	0.062	0.056	0.063	0.013	1.567
22	0.15	0.25	0.061	0.007	0.069	0.005	0.110
23	0.15	0.50	0.061	0.012	0.071	0.009	0.284
24	0.15	0.75	0.061	0.017	0.073	0.011	0.451
25	0.15	1.00	0.061	0.027	0.075	0.019	0.756
26	0.15	1.25	0.061	0.038	0.077	0.025	1.061
27	0.15	1.50	0.061	0.050	0.082	0.034	1.404
28	0.15	1.75	0.061	0.058	0.084	0.036	1.621
29	0.20	0.25	0.061	0.008	0.067	0.005	0.115
30	0.20	0.50	0.061	0.010	0.070	0.006	0.252
31	0.20	0.75	0.061	0.017	0.070	0.009	0.460
32	0.20	1.00	0.061	0.027	0.072	0.012	0.781
33	0.20	1.25	0.061	0.034	0.074	0.016	0.963

Gini Variable Importance							
simulation	ρ	β	Overall Mean	Overall Std.	Correlated Set Mean	Correlated Set Std.	Actual Predictor
34	0.20	1.50	0.061	0.046	0.073	0.019	1.301
35	0.20	1.75	0.062	0.056	0.076	0.024	1.578
36	0.25	0.25	0.061	0.008	0.068	0.004	0.132
37	0.25	0.50	0.061	0.010	0.069	0.005	0.229
38	0.25	0.75	0.061	0.017	0.072	0.010	0.489
39	0.25	1.00	0.061	0.028	0.076	0.013	0.817
40	0.25	1.25	0.062	0.041	0.078	0.019	1.184
41	0.25	1.50	0.061	0.050	0.080	0.022	1.429
42	0.25	1.75	0.062	0.060	0.081	0.025	1.718
43	0.30	0.25	0.061	0.008	0.067	0.005	0.092
44	0.30	0.50	0.061	0.010	0.075	0.008	0.229
45	0.30	0.75	0.061	0.017	0.084	0.013	0.434
46	0.30	1.00	0.061	0.026	0.090	0.017	0.729
47	0.30	1.25	0.061	0.038	0.103	0.026	1.038
48	0.30	1.50	0.061	0.045	0.108	0.033	1.248
49	0.30	1.75	0.061	0.056	0.117	0.040	1.537
50	0.35	0.25	0.061	0.009	0.069	0.006	0.105
51	0.35	0.50	0.061	0.011	0.078	0.009	0.220
52	0.35	0.75	0.061	0.018	0.090	0.017	0.450
53	0.35	1.00	0.061	0.028	0.112	0.029	0.719
54	0.35	1.25	0.061	0.042	0.135	0.044	1.084
55	0.35	1.50	0.061	0.051	0.138	0.050	1.342
56	0.35	1.75	0.062	0.064	0.155	0.065	1.666
57	0.40	0.25	0.061	0.009	0.070	0.007	0.095
58	0.40	0.50	0.061	0.012	0.085	0.010	0.282
59	0.40	0.75	0.061	0.018	0.096	0.015	0.452
60	0.40	1.00	0.061	0.029	0.114	0.025	0.734
61	0.40	1.25	0.061	0.039	0.137	0.033	0.990
62	0.40	1.50	0.061	0.049	0.151	0.043	1.263
63	0.40	1.75	0.061	0.058	0.157	0.052	1.510
64	0.45	0.25	0.061	0.008	0.071	0.004	0.105
65	0.45	0.50	0.061	0.010	0.084	0.010	0.181
66	0.45	0.75	0.061	0.019	0.116	0.021	0.394
67	0.45	1.00	0.061	0.026	0.132	0.026	0.578
68	0.45	1.25	0.061	0.037	0.164	0.043	0.800
69	0.45	1.50	0.061	0.045	0.180	0.050	1.013
70	0.45	1.75	0.061	0.058	0.220	0.064	1.254

Gini Variable Importance							
simulation	ρ	β	Overall Mean	Overall Std.	Correlated Set Mean	Correlated Set Std.	Actual Predictor
71	0.50	0.25	0.061	0.008	0.066	0.004	0.086
72	0.50	0.50	0.061	0.011	0.086	0.009	0.231
73	0.50	0.75	0.061	0.020	0.120	0.020	0.421
74	0.50	1.00	0.061	0.029	0.144	0.028	0.623
75	0.50	1.25	0.061	0.046	0.192	0.051	0.969
76	0.50	1.50	0.061	0.056	0.212	0.060	1.211
77	0.50	1.75	0.061	0.067	0.233	0.074	1.502
78	0.55	0.25	0.061	0.008	0.067	0.004	0.092
79	0.55	0.50	0.061	0.012	0.096	0.010	0.206
80	0.55	0.75	0.061	0.020	0.123	0.018	0.361
81	0.55	1.00	0.061	0.036	0.173	0.038	0.683
82	0.55	1.25	0.061	0.047	0.199	0.056	0.958
83	0.55	1.50	0.061	0.055	0.217	0.064	1.172
84	0.55	1.75	0.061	0.067	0.242	0.078	1.437
85	0.60	0.25	0.061	0.007	0.069	0.005	0.101
86	0.60	0.50	0.061	0.012	0.095	0.010	0.181
87	0.60	0.75	0.061	0.024	0.148	0.022	0.356
88	0.60	1.00	0.061	0.037	0.194	0.032	0.559
89	0.60	1.25	0.061	0.050	0.238	0.047	0.812
90	0.60	1.50	0.061	0.061	0.272	0.060	1.013
91	0.60	1.75	0.061	0.072	0.308	0.073	1.208
92	0.65	0.25	0.061	0.008	0.071	0.005	0.093
93	0.65	0.50	0.061	0.012	0.098	0.010	0.164
94	0.65	0.75	0.061	0.023	0.141	0.019	0.357
95	0.65	1.00	0.061	0.035	0.189	0.028	0.540
96	0.65	1.25	0.061	0.046	0.228	0.039	0.717
97	0.65	1.50	0.061	0.053	0.247	0.047	0.893
98	0.65	1.75	0.061	0.069	0.297	0.058	1.203
99	0.70	0.25	0.061	0.008	0.069	0.005	0.082
100	0.70	0.50	0.061	0.011	0.096	0.007	0.155
101	0.70	0.75	0.061	0.022	0.145	0.015	0.294
102	0.70	1.00	0.061	0.036	0.200	0.024	0.457
103	0.70	1.25	0.061	0.049	0.251	0.038	0.639
104	0.70	1.50	0.061	0.065	0.318	0.047	0.810
105	0.70	1.75	0.061	0.074	0.346	0.059	1.006
106	0.75	0.25	0.061	0.008	0.067	0.005	0.094
107	0.75	0.50	0.061	0.013	0.104	0.009	0.164
108	0.75	0.75	0.061	0.024	0.154	0.016	0.310

Gini Variable Importance							
simulation	ρ	β	Overall Mean	Overall Std.	Correlated Set Mean	Correlated Set Std.	Actual Predictor
109	0.75	1.00	0.061	0.041	0.218	0.032	0.545
110	0.75	1.25	0.061	0.054	0.272	0.044	0.709
111	0.75	1.50	0.061	0.068	0.323	0.057	0.922
112	0.75	1.75	0.061	0.080	0.368	0.064	1.068
113	0.80	0.25	0.061	0.008	0.070	0.005	0.083
114	0.80	0.50	0.061	0.013	0.105	0.008	0.140
115	0.80	0.75	0.061	0.023	0.152	0.014	0.279
116	0.80	1.00	0.061	0.039	0.221	0.024	0.440
117	0.80	1.25	0.061	0.056	0.287	0.037	0.614
118	0.80	1.50	0.061	0.066	0.325	0.045	0.734
119	0.80	1.75	0.061	0.079	0.377	0.062	0.883
120	0.85	0.25	0.061	0.008	0.066	0.004	0.074
121	0.85	0.50	0.061	0.013	0.105	0.006	0.156
122	0.85	0.75	0.061	0.025	0.161	0.015	0.269
123	0.85	1.00	0.061	0.040	0.223	0.016	0.410
124	0.85	1.25	0.061	0.055	0.285	0.028	0.570
125	0.85	1.50	0.062	0.070	0.346	0.040	0.711
126	0.85	1.75	0.061	0.079	0.380	0.045	0.845
127	0.90	0.25	0.061	0.007	0.060	0.004	0.075
128	0.90	0.50	0.061	0.012	0.104	0.007	0.150
129	0.90	0.75	0.061	0.029	0.180	0.013	0.251
130	0.90	1.00	0.061	0.045	0.248	0.019	0.395
131	0.90	1.25	0.061	0.057	0.301	0.026	0.491
132	0.90	1.50	0.062	0.072	0.364	0.031	0.639
133	0.90	1.75	0.061	0.085	0.414	0.044	0.777
134	0.95	0.25	0.061	0.007	0.065	0.003	0.071
135	0.95	0.50	0.061	0.012	0.104	0.007	0.140
136	0.95	0.75	0.061	0.030	0.188	0.010	0.238
137	0.95	1.00	0.061	0.045	0.250	0.017	0.353
138	0.95	1.25	0.061	0.060	0.317	0.020	0.429
139	0.95	1.50	0.061	0.076	0.384	0.025	0.553
140	0.95	1.75	0.061	0.091	0.447	0.032	0.692

APPENDIX B - R Code

```
#####################################################################
#######################                      ##########################
####################### SIMULATION CODE ##########################
#######################                      ##########################
#####################################################################

library(MASS)
library(randomForest)

#####################################################################

## Setting up dataset

## make.S is a function that makes a correlation matrix

make.S<-function(roe){

        Sigma<-diag(rep(1,40))

        for(i in 1:40){
                for(j in 1:40){
                        if(i!=j){Sigma[i,j]<-roe}
                }
        }

        Sigma
}

rm(x)
x<-numeric()
x.temp<-numeric()

## Creates the dataset

for(r in seq(0,.95,by=.05)){

        mu.vect<-runif(40,6,12)
        x.temp<-mvrnorm(100,mu.vect,make.S(r))
        x<-cbind(x,x.temp)

}

## classmaker generates the dependent variable
## (specify the column that the actual predictor comes from and
## the value for Beta)
```

```
classmaker<-function(column,Beta){

        eqtn<-(x[,column]-mean(x[,column]))/sqrt(var(x[,column]))
        logi<-exp(Beta*eqtn)/(1+exp(Beta*eqtn))
    cutoff<-runif(100)
        Y<-ifelse(cutoff<logi,0,1)
        Y<-as.factor(Y)

        Y
}

#####################################################################

## stimulation is the main function, this runs an entire simulation
## of size="num" with the predictor variable "col"
## for all values of Beta (b)

stimulation<-function(num,col){

n.sim<-num
column<-col
b<-c(.25,.5,.75,1,1.25,1.5,1.75)

## setting up matrices to store results

sim.imp<-matrix(ncol=length(b),nrow=800)
sim.imp.G<-matrix(ncol=length(b),nrow=800)
OOB<-matrix(ncol=length(b),nrow=n.sim)

## outer loop runs through values of Beta

for(j in 1:length(b)){

        big.I<-matrix(ncol=n.sim,nrow=800)
        big.I.G<-matrix(ncol=n.sim,nrow=800)
        oob<-numeric()

        ## inner loop runs through number of simulations "num"

        for(i in 1:n.sim){

        ## creates dependent variable then runs random forest

        Y<-classmaker(column,b[j])
        F<-randomForest(x,Y,importance=TRUE,ntree=2500)

        ## saving both types of importance measures and out-of-bag error

        big.I[,i]<-importance(F)[,3]
        big.I.G[,i]<-importance(F)[,4]
        oob[i]<-mean(F$err.rate[,1])
```

```
        }

        ## calculating means of importance measures over simulations

        sim.imp[,j]<-apply(big.I,1,mean)
        sim.imp.G[,j]<-apply(big.I.G,1,mean)
        OOB[,j]<-oob

}

## summarizing results of simulations
## (means and std.dev. of all variable importance measures)
## (means and std.dev. of correlated set variable importance measures)
## (values of actual predictor importance measures)

sim.mean<-apply(sim.imp,2,mean)
sim.std<-sqrt(apply(sim.imp,2,var))
sim.mean.grp<-apply(sim.imp[(column+1):(column+39),],2,mean)
sim.std.grp<-sqrt(apply(sim.imp[(column+1):(column+39),],2,var))
sim.pred.imp<-sim.imp[column,]

sim.mean.G<-apply(sim.imp.G,2,mean)
sim.std.G<-sqrt(apply(sim.imp.G,2,var))
sim.mean.grp.G<-apply(sim.imp.G[(column+1):(column+39),],2,mean)
sim.std.grp.G<-sqrt(apply(sim.imp.G[(column+1):(column+39),],2,var))
sim.pred.imp.G<-sim.imp.G[column,]

## listing output

list(sim.mean=sim.mean,sim.std=sim.std,sim.mean.grp=sim.mean.grp,
sim.std.grp=sim.std.grp,sim.pred.imp=sim.pred.imp,sim.imp=sim.imp,
sim.mean.G=sim.mean.G,sim.std.G=sim.std.G,sim.mean.grp.G=sim.mean.grp.G,
sim.std.grp.G=sim.std.grp.G,sim.pred.imp.G=sim.pred.imp.G,sim.imp.G=sim.imp.G,
OOB=OOB)

}

#####################################################################

## this is the code to run all simulations
## for all correlations and Beta values

b<-c(.25,.5,.75,1,1.25,1.5,1.75)
where.to.save<-"C:/Documents and Settings/rkimes/Desktop/hope2.RData"

## record results after each call of stimulation function
## into correlation specific vectors and matrices

s<-stimulation(10,1)
s.0<-cbind(b,s$sim.mean,s$sim.std,s$sim.mean.grp,s$sim.std.grp,s$sim.pred.imp)
```

```
raw.imp.0<-s$sim.imp
s.0.G<-
cbind(b,s$sim.mean.G,s$sim.std.G,s$sim.mean.grp.G,s$sim.std.grp.G,s$sim.pred.imp.G)
raw.imp.0.G<-s$sim.imp.G
oob.0<-s$OOB
save.image(where.to.save)

s<-stimulation(100,41)
s.05<-cbind(b,s$sim.mean,s$sim.std,s$sim.mean.grp,s$sim.std.grp,s$sim.pred.imp)
raw.imp.05<-s$sim.imp
s.05.G<-
cbind(b,s$sim.mean.G,s$sim.std.G,s$sim.mean.grp.G,s$sim.std.grp.G,s$sim.pred.imp.G)
raw.imp.05.G<-s$sim.imp.G
oob.05<-s$OOB
save.image(where.to.save)

.
.
# continue for all predictors
.
.

s<-stimulation(100,721)
s.90<-cbind(b,s$sim.mean,s$sim.std,s$sim.mean.grp,s$sim.std.grp,s$sim.pred.imp)
raw.imp.90<-s$sim.imp
s.90.G<-
cbind(b,s$sim.mean.G,s$sim.std.G,s$sim.mean.grp.G,s$sim.std.grp.G,s$sim.pred.imp.G)
raw.imp.90.G<-s$sim.imp.G
oob.90<-s$OOB
save.image(where.to.save)

s<-stimulation(100,761)
s.95<-cbind(b,s$sim.mean,s$sim.std,s$sim.mean.grp,s$sim.std.grp,s$sim.pred.imp)
raw.imp.95<-s$sim.imp
s.95.G<-
cbind(b,s$sim.mean.G,s$sim.std.G,s$sim.mean.grp.G,s$sim.std.grp.G,s$sim.pred.imp.G)
raw.imp.95.G<-s$sim.imp.G
oob.95<-s$OOB
save.image(where.to.save)
```

```
##############################################################
##############################################################
################    Microarray Analysis Code    ##############
##############################################################
##############################################################



library(randomForest)
library(affy)
library(GO)
library(annotate)
library(goTools)
library(hgu133a2)

memory.limit(size=4000)

##############################################################


## Creating X matrix of expression summaries and response Y

X<-exprs(both.rma.wo)
X<-t(X)
progress<-read.csv("e:/thesis/HCCPheno2.csv")$Progression
Y<-as.factor(progress)

##############################################################

## Checking Microarray Quality

control<-grep("AFFX",geneNames(both.rma))[13:26]
cnames<-geneNames(both.rma)[control]
h3<-grep("_3",cnames)[1:3]
h5<-grep("_5",cnames)[1:3]

ratio35<-exprs(both.rma)[control[h3],]/exprs(both.rma)[control[h5],]
max(ratio35)

##############################################################

#### Growing random forest with all variables

F<-randomForest(X,Y,ntree=2500,importance=TRUE)

## Plotting importance values

I.g<-importance(F)[,4]
plot(1:ncol(X),I.g,xlab="Predictor Variables")
plot(density(I.g))
hist(log(I.g))
```

```
## Calculating cutoff value

cut<-mean(I.g)+3*sqrt(var(I.g)/ncol(X))

## Creating new X matrix with unimportant variables eliminated

X.2<-X[,I.g>cut]

#### Growing random forest after first elimination

F.2<-randomForest(X.2,Y,ntree=2500,importance=TRUE)

## Plotting importance values

I.g2<-importance(F.2)[,4]
plot(1:ncol(X.2),I.g2)
plot(density(I.g2))
hist(log(I.g2))

## Calculating cutoff value

cut2<-mean(I.g2)+3*sqrt(var(I.g2)/ncol(X.2))

## Creating new X matrix with unimportant variables eliminated

X.3<-X.2[,I.g2>cut2]

#### Growing random forest after second elimination

F.3<-randomForest(X.3,Y,ntree=2500,importance=TRUE)

## Plotting importance values

I.g3<-importance(F.3)[,4]
plot(1:ncol(X.3),I.g3)
plot(density(I.g3))
hist(log(I.g3))

## Calculating cutoff value

cut3<-mean(I.g3)+3*sqrt(var(I.g3)/ncol(X.3))

## Creating new X matrix with unimportant variables eliminated

X.4<-X.3[,I.g3>cut3]

#### Growing random forest after third elimination

F.4<-randomForest(X.4,Y,ntree=2500,importance=TRUE)
```

```
## Plotting importance values

I.g4<-importance(F.4)[,4]
plot(1:ncol(X.4),I.g4)
plot(density(I.g4))
hist(log(I.g4))

## Calculating cutoff value

cut4<-mean(I.g4)+3*sqrt(var(I.g4)/ncol(X.4))

## Creating new X matrix with unimportant variables eliminated

X.5<-X.4[,I.g4>cut4]

### Growing final random forest

F.5<-randomForest(X.5,Y,ntree=2500,importance=TRUE)

## Plotting importance values

I.g5<-importance(F.5)[,4]
plot(1:ncol(X.5),I.g5)
plot(density(I.g5))
hist(log(I.g5))

#### Forward Selection ####

OOB.f<-matrix(nrow=101,ncol=3)

## Adding one variable at a time the model

for(i in 2:102){

v<-names(sort(I.g5,decreasing=TRUE)[1:i])
Fv<-randomForest(X.5[,v],Y,ntree=2500,importance=TRUE)
OOB.f[i-1,]<-apply(Fv$err.rate,2,mean)}

## Plotting out-of-bag error

plot(2:102,OOB.f[,1],type="b")

#### Random forest with top 4 important variables

v4<-names(sort(I.g5,decreasing=TRUE)[1:4])
Fv4<-randomForest(X.5[,v4],Y,ntree=2500,importance=TRUE)


#####################################################################

## Calculating Out-of-bag error rates of the random forests
```

```
rbind(apply(F$err.rate,2,mean),apply(F.2$err.rate,2,mean),
apply(F.3$err.rate,2,mean),apply(F.4$err.rate,2,mean),
apply(F.5$err.rate,2,mean),apply(Fv4$err.rate,2,mean))


## names of most important genes

fin<-names(I.g5)
best.n<-names(sort(I.g5,decreasing=TRUE))



#####################################################################

## setup for Gene Ontology comparison

sig.genes<-list(L1=best.n,L2=geneNames(both.rma.wo))
MFendnode <- CustomEndNodeList("GO:0003674", rank=2)

## comparing molecular functions

GO.mf<-ontoCompare(sig.genes, probeType="hgu133a2", endnode=MFendnode,
goType="MF",plot=FALSE)
xlabels<-gsub("activity","",dimnames(GO.mf)[[2]])
par(las=2)
barplot(GO.mf,cex.names=0.6,beside=TRUE,names.arg=xlabels,legend.text=c("Final
Variable Set","GeneChip"),main="GO Molecular Function")

ratio.mf<-GO.mf[1,]/GO.mf[2,]
ratio1<-ratio.mf>1
molecule<-ratio.mf[ratio1]
names(sort(molecule,decreasing=TRUE))

par(las=2)
barplot(GO.mf[ratio1],cex.names=0.6,beside=TRUE,names.arg=xlabels[ratio1],legend.tex
t=c("Final Variable Set","GeneChip"),main="GO Molecular Function")

## comparing biological processes

GO.bp<-ontoCompare(sig.genes, probeType="hgu133a2", goType="BP",plot=FALSE)
xlabels<-gsub("activity","",dimnames(GO.bp)[[2]])
par(las=2)
barplot(GO.bp,cex.names=0.6,beside=TRUE,names.arg=xlabels,legend.text=c("Final
Variable Set","GeneChip"),main="GO Biological Process")

ratio.bp<-GO.bp[1,]/GO.bp[2,]
biology<-ratio.bp[ratio.bp>1]
sort(biology,decreasing=TRUE)

## comparing cellular components
```

```
GO.cc<-ontoCompare(sig.genes, probeType="hgu133a2", goType="CC",plot=FALSE)
xlabels<-gsub("activity","",dimnames(GO.cc)[[2]])
par(las=2)
barplot(GO.cc,cex.names=0.6,beside=TRUE,names.arg=xlabels,legend.text=c("Final
Variable Set","GeneChip"),main="GO Cellular Component")

ratio.cc<-GO.cc[1,]/GO.cc[2,]
cellcomp<-ratio.cc[ratio.cc>1]
sort(cellcomp,decreasing=TRUE)
```

APPENDIX C – List of Genes in S*

Gene ID	Affymetrix ID	Symbol	UniGene Cluster	Chromosome	Chromose Band
6155	200025_s_at	RPL27	Hs.513705	17	17q21.1-q21.2
3927	200618_at	LASP1	Hs.334851	17	17q11-q21.3
10097	200727_s_at	ACTR2	Hs.393201	2	2p14
6160	200963_x_at	RPL31	Hs.469473	2	2q11.2
8396	201080_at	PIP5K2B	Hs.260603	17	17q12
5111	201202_at	PCNA	Hs.147433	20	20pter-p12
4122	202032_s_at	MAN2A2	Hs.116459	15	15q26.1
6566	202235_at	SLC16A1	Hs.75231	1	1p12
2962	202354_s_at	GTF2F1	Hs.68257	19	19p13.3
84747	202365_at	MGC5139	Hs.127610	12	12q24.31
5054	202628_s_at	SERPINE1	Hs.414795	7	7q21.3-q22
51406	202882_x_at	NOL7	Hs.549161	6	6p23
54541	202887_s_at	DDIT4	Hs.523012	10	10pter-q26.12
9903	203068_at	KLHL21	Hs.7764	1	1p36.31
8604	203340_s_at	SLC25A12	Hs.470608	2	2q24
10106	203445_s_at	CTDSP2	Hs.524530	12	12q13-q15
10475	203568_s_at	TRIM38	Hs.202510	6	6p21.3
8449	203694_s_at	DHX16	Hs.485060	6	6p21.3
88	203861_s_at	ACTN2	Hs.498178	1	1q42-q43
88	203862_s_at	ACTN2	Hs.498178	1	1q42-q43
10681	204000_at	GNB5	Hs.155090	15	15q21.2
864	204198_s_at	RUNX3	Hs.170019	1	1p36
1119	204266_s_at	CHKA	Hs.77221	11	11q13.2
9844	204513_s_at	ELMO1	Hs.304578	7	7p14.1
2289	204560_at	FKBP5	Hs.407190	6	6p21.3-21.2
22943	204602_at	DKK1	Hs.40499	10	10q11.2
10468	204948_s_at	FST	Hs.9914	5	5q11.2
5324	205372_at	PLAG1	Hs.14968	8	8q12
7850	205403_at	IL1R2	Hs.25333	2	2q12-q22
29901	205449_at	SAC3D1	Hs.23642	11	11q13.1
4246	205979_at	SCGB2A1	Hs.97644	11	11q13
9837	206102_at	PSF1	Hs.360033	20	20p11.21
7098	206271_at	TLR3	Hs.29499	4	4q35
9971	206340_at	NR1H4	Hs.282735	12	12q23.1
5444	206344_at	PON1	Hs.370995	7	7q21.3
5444	206345_s_at	PON1	Hs.370995	7	7q21.3
1621	206450_at	DBH	Hs.223858	9	9q34
7090	206472_s_at	TLE3	Hs.287362	15	15q22
8464	206506_s_at	SUPT3H	Hs.368325	6	6p21.1-p21.3

Gene ID	Affymetrix ID	Symbol	UniGene Cluster	Chromosome	Chromose Band
28513	206898_at	CDH19	Hs.42771	18	18q22-q23
27109	206992_s_at	ATP5S	Hs.438489	14	14q21.3
10468	207345_at	FST	Hs.9914	5	5q11.2
NA	207472_at	NA	NA	NA	NA
9173	207526_s_at	IL1RL1	Hs.66	2	2q12
54941	207735_at	RNF125	Hs.272800	18	18q12.1
80108	207757_at	ZFP2	Hs.484328	5	5q35.3
6175	208856_x_at	RPLP0	Hs.448226	12	12q24.2
56339	209265_s_at	METTL3	Hs.168799	14	14q11.1
10111	209349_at	RAD50	Hs.242635	5	5q31
9169	209376_x_at	SFRS2IP	Hs.210367	12	12q13.11
5414	210657_s_at	4-Sep	Hs.287518	17	17q22-q23
8725	211563_s_at	C19orf2	Hs.466391	19	19q12
8837	211862_x_at	CFLAR	Hs.390736	2	2q33-q34
6175	211972_x_at	RPLP0	Hs.448226	12	12q24.2
23265	212034_s_at	EXOC7	Hs.533985	17	17q25.1
1457	212072_s_at	CSNK2A1	Hs.446484	20	20p13
6430	212266_s_at	SFRS5	Hs.166975	14	14q24
3097	212641_at	HIVEP2	Hs.510172	6	6q23-q24
89927	212736_at	C16orf45	Hs.460095	16	16p13.11
84986	212738_at	ARHGAP19	Hs.80305	10	10q24.1
26268	212987_at	FBXO9	Hs.216653	6	6p12.3-p11.2
91949	213190_at	COG7	Hs.185807	16	16p12.1
7148	213451_x_at	TNXB	Hs.42853	6	6p21.3
898	213523_at	CCNE1	Hs.244723	19	19q12
387032	213625_at	ZNF307	Hs.44720	6	6p21.33-p21.31
6224	214003_x_at	RPS20	Hs.8102	8	8q12
3013	214522_x_at	HIST1H2AD	Hs.534319	6	6p21.3
63982	215241_at	TMEM16C	Hs.91791	11	11p14.3
1827	215253_s_at	DSCR1	Hs.282326	21	21q22.1-q22.2 21q22.12
7035	215447_at	TFPI	Hs.516578	2	2q31-q32.1
85021	215922_at	REPS1	Hs.334603	6	NA
7148	216333_x_at	TNXB	Hs.42853	6	6p21.3
4499	217546_at	MT1M	Hs.188518	16	16q13
10135	217738_at	PBEF1	Hs.489615	7	7q22.3
10135	217739_s_at	PBEF1	Hs.489615	7	7q22.3
55740	217820_s_at	ENAH	Hs.497893	1	1q42.12
28972	217927_at	SPCS1	Hs.11125	3	3p21.1
22822	217997_at	PHLDA1	Hs.484885	12	12q15
22822	217998_at	PHLDA1	Hs.484885	12	12q15
8566	218019_s_at	PDXK	Hs.284491	21	21q22.3
29088	218027_at	MRPL15	Hs.18349	8	8q11.2-q13
29964	218233_s_at	C6orf49	Hs.525899	6	6p21.31
80017	218298_s_at	C14orf159	Hs.309849	14	14q32.12

Gene ID	Affymetrix ID	Symbol	UniGene Cluster	Chromosome	Chromose Band
51053	218350_s_at	GMNN	Hs.234896	6	6p22.2
55813	218715_at	C17orf40	Hs.211828	17	17q11.2
54020	218928_s_at	SLC37A1	Hs.473918	21	21q22.3
80304	219120_at	FLJ21945	Hs.24624	2	2p23.3
55039	219299_at	TRMT12	Hs.9925	8	8q24.13
79818	219741_x_at	ZNF552	Hs.129691	19	19q13.43
7767	220019_s_at	ZNF224	Hs.549077	19	19q13.2
23225	220035_at	NUP210	Hs.475525	3	3p25.2-p25.1
50809	220633_s_at	HP1BP3	Hs.142442	1	1p36.12
54808	220774_at	DYM	Hs.162996	18	18q12-q21.1
140545	220991_s_at	RNF32	Hs.490715	7	NA
51174	221326_s_at	TUBD1	Hs.463638	17	17q23.2
58528	221524_s_at	RRAGD	Hs.485938	6	6q15-q16
7187	221571_at	TRAF3	Hs.510528	14	14q32.32
6477	221834_at	SIAH1	Hs.295923	16	16q12
51312	221920_s_at	SLC25A37	Hs.122514	8	8p21.2
57570	221952_x_at	TRMT5	Hs.380159	14	14q23.1
26137	222357_at	ZBTB20	Hs.122417	3	3q13.2
6453	35776_at	ITSN1	Hs.160324	21	21q22.1-q22.2

VITA

Ryan Kimes was born on August 4, 1981 in Panorama City, CA. He went to Santa Ynez Valley Union High School in Santa Ynez, CA. He graduated with a B.S. in Statistics from California Polytechnic State University San Luis Obispo in 2004. While at Cal Poly, Ryan served as president of the Beta Theta Pi fraternity and made the President's Honor List in 2001-2002. Ryan will graduate from Virginia Commonwealth University with a M.S. in Biostatistics in May 2006. He has served as a statistical consultant for VCU Technology Services and was recently inducted into the Phi Kappa Phi honor society. He will be moving back to sunny California to pursue career interests in gene expression analysis and saline wave navigation.