Fuzzy Forests: Extending Random Forests for Correlated, High-Dimensional Data

Daniel Conn
UCLA SPH, Biostatistics

Tuck Ngun
Department of Human Genetics David Geffen School of Medicine, UCLA

Gang Li
UCLA SPH, Biostatistics

Christina M. Ramirez
UCLA SPH, Biostatistics

Abstract

In this paper we introduce fuzzy forests, a novel machine learning algorithm for ranking the importance of features in high-dimensional classification and regression problems. Fuzzy forests is specifically designed to provide relatively unbiased rankings of variable importance in the presence of highly correlated features, especially when $p >> n$. We introduce our implementation of fuzzy forests in the R package, **fuzzyforest**. Fuzzy forests works by taking advantage of the network structure between features. First, the features are partitioned into separate modules such that the correlation within modules is high and the correlation between modules is low. The package **fuzzyforest** allows for easy use of Weighted Gene Coexpression Network Analysis (WGCNA) to form modules of features such that the modules are roughly uncorrelated. Then recursive feature elimination random forests (RFE-RFs) are used on each module, separately. From the surviving features, a final group is selected and ranked using one last round of RFE-RFs. This procedure results in a ranked variable importance list whose size is pre-specified by the user. The selected features can then be used to construct a predictive model.

Keywords: Random Forests, WGCNA, machine learning, R, networks, $p >> n$, big data, variable selection, variable importance, variable ranking.

1. Introduction

In the era of high-throughput technologies such as multi-color flow cytometry and next generation sequencing, high dimensional data has become increasingly common in biomedical research. However, the ability to generate data has vastly outpaced our ability to analyze it. In the biomedical sciences as well as the 'Omics fields it is common for the number of parameters to be much larger than the number of observations, the so-called $p >> n$ problem. This problem is exacerbated by the fact that the features are often highly correlated and the correlation structure is often unknown a priori.

Identifying important features in this situation has been an area of intense research within the

statistics and machine learning community. While model based feature selection algorithms such as the LASSO or SCAD may detect important features in the presence of correlation (Raskutti *et al.* 2010), this comes at the cost of making parametric assumptions that may not hold in practice. If $p >> n$, the LASSO can select at most n features. In the setting where there are groups of highly correlated features, the LASSO has a tendency to arbitrarily pick one of the features from a group and shrink the others towards zero (Bondell and Reich 2008).

Random forests are a popular ensemble machine learning algorithm. Random forests are non-parametric, non-linear, embarrassingly parallelizable, easy to implement, and have been described as one of the best "off-the-shelf" classifiers (Dua *et al.* 2014). Random forest variable importance measures (VIMs) offer a flexible alternative to model based feature selection algorithms (Breiman 2001). While random forest VIMs have demonstrated the ability to accurately capture the true importance of features in settings where the features are independent, it is well-known that random forest VIMs are biased when features are correlated with one another (Strobl *et al.* (2007, 2008); Nicodemus and Malley (2009)).

Fuzzy forests handle correlated features by taking a piecewise approach. We first estimate the network structure of the data and partition the set of features into distinct modules such that the correlation within each module is high and the correlation between modules is low. In this regard, we utilize the functionality of Weighted Gene Coexpression Network Analysis, a rigorous framework for detecting correlation networks (Zhang and Horvath 2005). We then use Recursive Feature Elimination Random Rorests (RFE-RFs) as in (Díaz-Uriarte and De Andres 2006) to select the most important features from each module. One final RFE-RF is then applied to the remaining features, selecting and ranking the most important ones.

The article is organized as follows. In section 2 of this article, we briefly review random forests, WGCNA, and introduce the fuzzy forests algorithm. In section 3, we introduce the R package **fuzzyforest**. In section 4, we conduct simulations comparing fuzzy forests to random forests. In section 5, we use fuzzy forests to determine which immunologic factors are important in determining a patient's response to HIV infection. Section 6 ends the article with a discussion and summary of our results.

2. Variable Importance Measures and the Fuzzy Forests Algorithm

2.1. Variable Importance Measures

In this section, we introduce basic notation and discuss variable importance measures. We assume that our data comes in the form of n independently and identically distributed (iid) pairs $(X, Y) \sim G_{(X,Y)}$. Here, X is a p dimensional feature vector and Y is a scalar outcome. Let $X_i^{(v)}$ denote the value of the vth feature for the ith subject and let $X_i = (X_i^{(1)}, \ldots, X_i^{(p)})$ be the feature vector for the ith subject. Signifying the set of values for feature v across all n subjects, we set $X^{(v)} = (X_1^{(v)}, \ldots, X_n^{(v)})$. Finally, the marginal distributions of X and $X^{(v)}$ are written as G_X and $G_{X^{(v)}}$, respectively.

In the case of both classification and regression, we are interested in modeling the conditional mean of Y given a feature vector X. We denote this conditional mean as $E[Y|X]$ or $f(X)$. In the case of regression, we assume that $Y|X$ has distribution $f(X) + \epsilon$, where the ϵ are

independent of X and iid with variance σ^2. In the regression setting, Y is continuous. In binary classification, if Y is restricted to take the value 0 or 1 then $Y|X$ is a Bernoulli trial with mean $E[Y|X = x] = P(Y = 1|X = x)$. For regression, a prediction for a new observation X_{new} would obtained by evaluating the conditional mean at X_{new}: $f(X_{new})$. In the case of binary classification, the predicted outcome for a new observation would be 1 if $f(X_{new}) = P(Y = 1|X = X_{new}) > 0.5$, and 0 otherwise. Note that random forests can easily be generalized to handle the case of multi-class classification.

If the goal were to predict a new outcome Y based off of the features, X, a good estimate of $f(X)$ would suffice. The problem of feature selection requires more than a "black box" estimate of $f(X)$. It requires an understanding of how $f(X)$ depends on each individual feature.

If p is low dimensional ($p = 1, 2$), we can simply plot our estimate of $f(X)$ to understand how it varies as function of X. On the other hand, if p is moderate or large, the estimate of $f(X)$ may be difficult to interpret. This problem of interpretability may be alleviated by assuming $f(X)$ has a specific parametric form such that $f_\gamma(X)$ is known up to a finite dimensional parameter γ. In the case of linear regression, where $f_\gamma(X_i) = \sum_{v=1}^{p} \gamma_v X_i^{(v)}$, γ is a vector of regression coefficients and we may measure the importance of one feature versus another by examining the absolute magnitude of their corresponding coefficients (assuming the features have all been standardized).

However, we rarely believe that $f_\gamma(X) = f(X)$ for some γ. Rather, $f_\gamma(X)$ is often thought of as a parametric approximation to $f(X)$. Unfortunately, this parametric approximation may fail to capture salient characteristics of $f(X)$ for a variety of reasons. Notably, $f_\gamma(X)$ might miss important interactions between features, or, in the case of the above linear regression model, the true $f(X)$ may be nonlinear in such a way that the best linear approximation fails to capture. In contrast, random forests are non-linear and non-parametric. Therefore, the resulting random forest VIMs, defined below, naturally take interactions and non-linear structure into account.

2.2. An Introduction to Random Forests

Random forests is a popular ensemble method that has been applied in the setting of both classification and regression. The random forests algorithm works by combining the predictions of an ensemble of classification or regression trees. Each tree is grown on a separate bootstrap sample of the data. The number of trees grown in this manner is denoted as *ntree*. The subjects that are not selected in a particular bootstrap sample are said to be "out of bag." Note that roughly one third of observations will be out of bag for each tree. These samples play the important role of serving as a test set for each tree, allowing the user to obtain estimates of the prediction error that are not overly optimistic.

Call the kth tree $\hat{f}_k(X)$. In the case of regression trees, $\hat{f}(X) = \frac{1}{ntree} \sum_{k=1}^{ntree} \hat{f}_k(X)$. In the case of classification, $\hat{f}(X)$ is the majority vote of the *ntree* predictions given by $\hat{f}_k(X)$. Each regression tree, by itself, may be highly unstable, leading to high variability estimates of $f(X)$, however, by averaging multiple trees over many bootstrap samples, the variance of our estimate for $f(X)$ may be significantly reduced. The algorithm described thus far is known as bagging (bootstrap-aggregating). This algorithm is a special case of random forests.

A further element of randomness is introduced by random forests. Before a node in a particular

tree is split, a subset of features is chosen at random. The best splitting rule, involving only these randomly selected features, is then used to split the node. The number of randomly selected features at each stage is commonly called $mtry$. High values of $mtry$ tend to lead to just a few important features getting selected at the majority of nodes. If $mtry = p$, then random forests are equivalent to bagging. Lower values of $mtry$ allow more features to play a role in the estimation $f(X)$. In the case of regression, a common default value of $mtry$ is $\lfloor p/3 \rfloor$. In the case of classification $\sqrt{p}$ is common choice.

Random forest VIMs are obtained by testing how predictive accuracy suffers when the values of an individual feature are randomly permuted. Suppose a particular feature is important in determining the outcome, Y. Permuting the values of this feature destroys its relationship with the outcome. Because this important relationship has been destroyed, there should be a subsequent decrease in predictive accuracy if a random forest is fit to this permuted data set. If there was no relationship to begin with, the predictive accuracy of the random forest fit to the permuted data should remain unaffected. VIMs measure the average decline in predictive performance for each feature across multiple trees.

The VIM for the vth feature is calculated as follows below. Let $OOB_k \subset \{1, \ldots, n\}$ be the indices for the out of bag sample from the kth tree. Let $\pi_k = (\pi_{k1}, \ldots, \pi_{kn})$ be a random permutation of OOB_k and let $\tilde{X}_i = (X_i^{(1)}, \ldots, X_{\pi_{ki}}^{(v)}, \ldots, X_i^{(p)})$ be the feature vector for the ith subject where the vth feature has been permuted. In the case of regression, the variable importance of the vth feature from the kth tree is defined as

$$\widehat{VIM}_k(v) = \frac{\sum_{i \in OOB_k} (y_i - \hat{f}_k(\tilde{X}_i))^2 - (y_i - \hat{f}_k(X_i))^2}{|OOB_k|} \tag{1}$$

The variable importance for the entire random forest is defined as

$$\widehat{VIM}(v) = \frac{\sum_{k=1}^{ntree} \widehat{VIM}_k(v)}{ntree} \tag{2}$$

2.3. A Brief Review of WGCNA

In biology, statistical network models play a significant role in uncovering important regulatory mechanisms or processes. WGCNA, first developed to detect networks of highly correlated genes, has seen great success in many such biological applications. The R package **WGCNA** is a robust and well-documented implementation of the WGCNA framework (Langfelder and Horvath 2008). We expect that researchers already familiar with **WGCNA** will easily adopt the fuzzy forests algorithm and we expect that newcomers to WGCNA will be able to make good use of the help file in **fuzzyforest** as well as **WGCNA**'s fine documentation and tutorials.

To construct a network, we first need a similarity function. This similarity function is often closely related to the Pearson correlation. We then weight them to emphasize strong correlations and punish weak ones by taking the absolute value and raising to the power β. These weighted correlations measure the connection strength, c between the features in the network. By adding up these connection strengths for each feature, we get the connectivity. This describes how strongly each feature v is connected to the other features in the network. The adjacency matrix is the matrix of connection strengths. Next, we identify groups of features with high topological overlap. A pair of features has high topological overlap if both variables are strongly connected to the same group of features. After calculating the topological

overlap for each pair of features, we use hierarchical clustering to identify modules of densely interconnected features. The correlation of features within each module is high, while the correlation of features between modules is low.

Formally, the user first specifies a similarity function $s_{uv} = S(X^{(u)}, X^{(v)})$ for features u and v, taking values between 0 and 1. Both unsigned and signed networks are possible. If the features are continuous, the most common choice of similarity function is $|Corr(X^{(u)}, X^{(v)})|$ or $\frac{1+Corr(X^{(u)},X^{(v)})}{2}$ according to whether the network is unsigned or signed, respectively (Zhang and Horvath 2005).

This similarity matrix is then transformed into an adjacency matrix $A = [a_{uv}]$. The adjacency function determine how similarities translate into properties of the network. For example, in a hard-thresholded network, where nodes are either connected or unconnected, the adjacency function determines how high the correlation has to be in order for two nodes to be connected.

This hard threshold function, denoted by $signum(s_{uv}, \tau)$, where τ is defined to be the threshold, is the simplest choice of adjacency function. If $s_{uv} \geq \tau$ then $a_{uv} = signum(s_{uv}, \tau) = 1$, otherwise $a_{uv} = 0$. Nodes are either then classified as connected or unconnected. In practice, a soft-thresholded network is often more plausible than a hard-thresholded one. The power function $a_{uv} = s_{uv}^{\beta}$ is common choice of soft-thresholding adjacency function. Large values of β yield behavior closer to a hard-thresholded network. Setting $\beta = 1$ is equivalent to using the similarity function alone. Once an adjacency function is calculated, a hierarchical clustering tree algorithm is used to define the clusters of features.

It is common to apply this hierarchical clustering algorithm to the topological overlap matrix rather than the adjacency matrix. The topological overlap between two nodes is defined as

$$\omega_{uv} = \frac{q_{uv} + a_{uv}}{min\{c_u, c_v\} + 1 - a_{uv}} \tag{3}$$

where $q_{uv} = \sum_{r=1}^{p} a_{ir} a_{rj}$ and $c_i = \sum_{r=1}^{p} a_{ir}$ (Horvath 2011). The topological overlap between two nodes can be high even if a_{uv} is low. This occurs when the two nodes are strongly connected to the same set of nodes. Use of topological overlap rather than the adjacencies may lead to more distinct modules (Zhang and Horvath 2005).

In many biological contexts, it is suspected that only a few features are highly connected. This prior knowledge leads to the scale-free criterion for determining which value of q to select. A network is said to have a generalized scale-free topology if $c(z) \propto z^{\theta}$, where $c(z)$ is the frequency function for the connectivity and θ is non-negative real number. $\log_{10}(c(z)) \propto \log_{10}(z)$ (Zhang and Horvath 2005). If the scale-free topology criterion is suspected to hold, one should select a value of β such that the R^2 between $\log_{10}(c(z))$ and $\log_{10}(z)$ is high.

2.4. The Fuzzy Forests Algorithm

The fuzzy forests algorithm is an extension of random forests designed to obtain less biased variable importance rankings in the presence of correlated features. In the following section, we describe the motivation behind fuzzy forests and explain why it provides relatively unbiased rankings of VIMs. In this section, we describe the algorithm.

The fuzzy forests algorithm reduces the parameter space in two steps: a screening step and a selection step. The screening step works in a piecewise fashion to screen out unimportant features once the features have been assigned to modules. The screening step takes as input

a module or partition of the features such that the correlation within each module is high. Our package, **fuzzyforest**, facilitates the use of WGCNA to determine the modules although, it is possible to use alternative methods to partition the features, if this is known a priori. Denote this partitioning of the features by the set $P = \{P_1, \ldots, P_m\}$. Let $p_l = |P_l|$ so that $\sum_{l=1}^{m} p_l = p$.

The screening step operates independently on each partition. For each element of the partition P_l, Recursive Feature Elimination Random Forests (RFE-RF) is used to screen out unimportant features. Starting with all features in partition P_l, a random forest is fit and the least important features are then eliminated. Call the reduced set of features in P_l, after the first random forest, $P_l^{(1)}$. For example, the features with VIM in the bottom 25% might be dropped at each step. A 2nd random forest is then fit using features in $P_l^{(1)}$. The least important features from this latest random forest are then eliminated leading to a further reduced set of features $P_l^{(2)} \subset P_l^{(1)} \subset P_l$. The subset obtained after iteration t is denoted as $P_l^{(t)}$ and let $p_l^{(t)}$ be the number of features in $P_l^{(t)}$. Features are eliminated in this manner until a user-specified stopping criteria is reached. For example, features may be eliminated until 5% of the original features in P_l remain.

The user must specify a few tuning parameters at the screening step. First, the user must specify how many features are to be dropped after each step of the RFE-RF. We call this fraction the $drop\_fraction$. The user must also specify a stopping criteria. In **fuzzyforest** the user specifies what percentage of the original p_l features to retain. This percentage is called the $keep\_fraction$. The first time the number of features drops below $keep\_fraction * p_l$, the RFE-RF stops and the top $\lfloor keep\_fraction * p_l \rfloor$ features are selected. More precisely, for the first iteration t such that $p_l^{(t)} < keep\_fraction * p_l$, we retain the top $\lfloor keep\_fraction * p_l \rfloor$ features from $P_l^{(t-1)}$.

For each random forest RFE-RF, $mtry$ and $ntree$ must be appropriately selected. Since the number of features varies across random forests, $mtree$ and $ntree$ must be a function of the current number of features. Suppose we are at iteration t and are about fit a random forest to obtain $P_l^{(t+1)} \subset P_l^{(t)}$. In the case of regression, **fuzzyforest** sets $mtry = \sqrt{p_l^{(t)}} * mtry\_factor$. For classification **fuzzyforest** sets $mtry = \left\lfloor p_l^{(t)}/3 \right\rfloor * mtry\_factor$. In both cases, $mtry\_factor$ must be pre-specified by the user, with the default being 1. The parameter $ntree$ must be set high enough to be able to pick up the effects of important variables, however if $ntree$ is set too high, the iterative series of random forests could increase the run time. The package **fuzzyforest** sets $ntree = max(min\_ntree, p_l^{(t)} * ntree\_factor)$.

The selection step consists of one last RFE-RF to allow for interactions between modules. This RFE-RF is applied to all features that have been selected at the screening step. Note that a separate choice of $drop\_fraction$, $mtry\_factor$, min_tree, and $ntree\_factor$ may be used. In the package **fuzzyforest**, $keep\_fraction$ is implicitly defined by user, as the user specifies how many features they would like in the final selection step.

2.5. Motivation for Fuzzy Forests Algorithm

The selection step of fuzzy forests is motivated by the following observations concerning the theoretical properties of VIMs. Permutation VIMs provide a means of summarizing the importance of individual features without making parametric assumptions. In the case of

regression, the random forest permutation VIM of feature v estimates the following parameter:

$$VIM(v) = E(f(X_i^{(1)}, \ldots, X_i^{(v)}, \ldots, X_i^{(p)}) - f(X_i^{(1)}, \ldots, \tilde{X}_i^{(v)}, \ldots, X_i^{(p)}))^2. \tag{4}$$

The above expression deserves further explanation. First, note that the expression is the same for all choices of index, i, because the (X_i, Y_i) are iid with distribution $G_{(X,Y)}$. Next note that f is fixed and the expectation is with respect to the random variables $X_i = (X_i^{(1)}, \ldots, X_i^{(v)}, \ldots, X_i^{(p)})$ and $\tilde{X}_i^{(v)}$. The random vector X_i has distribution G_X and $\tilde{X}_i^{(v)}$, generated independently of X_i, has distribution $G_{X^{(v)}}$. Here, $\tilde{X}_i^{(v)}$ can be thought of as an independently generated realization from $X_i^{(v)}$. If the value of $f(X_i)$ changes greatly when $X_i^{(v)}$ is replaced by $\tilde{X}_i^{(v)}$, it implies that the vth feature is important. In the case where $f_\gamma(X) = \sum_{v=1}^{p} \gamma_v X^{(v)}$ is a linear model, with standardized features ($var(X_i^{(v)}) = 1$), $VIM(v) = \gamma_v^2$.

This form of the VIM is given in a slightly different form in Gregorutti *et al.* (2013) and Zhu *et al.* (2012). Gregorutti *et al.* (2013) present a similar expression for the case of classification. These authors also discuss conditions under which the estimate of the permutation VIM derived from random forests is consistent.

Let $G_{P^{(l)}}$ denote the joint distribution of the features in partition $P^{(l)}$ and let $X^{P^{(l)}} \sim G_{P^{(l)}}$. In general, the conditional expectation, $E[A|B]$, of one random variable A with respect to another random variable, B, is defined as the function $h(B)$ that minimizes $E[(A - h(B))^2]$ or, written more compactly, $\text{argmin}_h E[(A - h(B))^2]$. When random forests are fit using only the features in module $P^{(l)}$, the estimated regression function converges to

$$\begin{aligned}
\text{argmin}_h E[(Y - h(X^{P^{(l)}}))^2] &= \text{argmin}_h E[(f(X) + \epsilon - h(X^{P^{(l)}}))^2] && (5)\\
&= \text{argmin}_h E[\epsilon^2 + 2\epsilon(f(X) - h(X^{P^{(l)}})) + (f(X) - h(X^{P^{(l)}}))^2] && (6)\\
&= \text{argmin}_h E[2\epsilon(f(X) - h(X^{P^{(l)}})) + (f(X) - h(X^{P^{(l)}}))^2] && (7)\\
&= \text{argmin}_h E[(f(X) - h(X^{P^{(l)}}))^2] && (8)\\
&= E[f(X)|X^{P^{(l)}}]. && (9)
\end{aligned}$$

Note that the $E[2\epsilon(f(X) - h(X^{P^{(l)}}))] = 0$ because ϵ is indpendent of X and has mean 0.

Suppose that features in separate modules $X^{P^{(1)}}, \ldots, X^{P^{(m)}}$ are independent and suppose that $f(X) = \sum_{j=1}^{m} f_j(X^{P^{(j)}})$. The regression function $f(X)$ allows for interactions within modules and no interactions between modules. We now demonstrate that if we fit a random forest using only the features in $P^{(l)}$, we are no longer estimating $E[Y|X] = f(X)$, instead we are estimating

$$E[f(X)|X^{P^{(l)}}] = \sum_{j=1}^{m} E[f_j(X^{P^{(j)}})|X^{P^{(l)}}] = f_l(X^{P^{(l)}}) + \sum_{j \neq l}^{m} E_{X_{P^{(l)}}}[f_j(X^{P^{(j)}})]. \tag{10}$$

As a result, the VIMs obtained by fitting a separate random forest to each module $P^{(l)}$, are equal to the VIMs obtained by fitting a random forest to the full set of features.

This is seen by the following argument. Thus,

$$E[Y|X^{P^{(l)}}] = \text{argmin}_h E[(Y - h(X^{P^{(l)}}))^2] \tag{11}$$

$$= \text{argmin}_h E[\{(Y - f(X)) - (h(X^{P^{(l)}}) - f(X))\}^2] \tag{12}$$

.

This last term equals:

$$\text{argmin}_h \{E[(Y - f(X))^2] - 2E[(Y - f(X))(h(X^{P^{(l)}}) - f(X))] + E[(h(X^{P^{(l)}}) - f(X))^2]\}. \tag{13}$$

Now, the first of the above expectations does not depend on g. The second expectation equals 0:

$$E[(Y - f(X))(h(X^{P^{(l)}}) - f(X))] = E[E[(Y - f(X))(h(X^{P^{(l)}}) - f(X))|X]] \tag{14}$$

$$= E[(h(X^{P^{(l)}}) - f(X))E[(Y - f(X))|X]] \tag{15}$$

$$= 0. \tag{16}$$

This leaves only the third expectation remaining. Thus, $E[Y|X^{P^{(l)}}] = \text{argmin}_h E[(h(X^{P^{(l)}}) - f(X))^2]$. By the definition of conditional expectation, this last term equals $E[f(X)|X^{P^{(l)}}]$. Note that by the independence of the modules, we have $E[f_j(X^{P^{(j)}})|X^{P^{(l)}}] = E_{X_{P^{(l)}}}[f_j(X^{P^{(j)}})]$ for all $j \neq i$. This yields equation (10).

Suppose feature v is in partition $P^{(l)}$. The variable importance obtained by fitting a random forest to only those features in $P^{(l)}$ is estimating the following quantity:

$$VIM^*(v) = E(f_l(X_i^{(l_1)}, \ldots, X_i^{(v)}, \ldots, X_i^{(l_m)}) - f_l(X_i^{(l_1)}, \ldots, \tilde{X}_i^{(v)}, \ldots, X_i^{(l_m)}))^2. \tag{17}$$

Here, $X_i^{l_k}$ is the kth element of partition $P^{(l)}$. As in equation (4), $X^{(v)}$ and $\tilde{X}^{(v)}$ are iid from $G_{X^{(v)}}$. We see from this equation that $VIM^*(v) = VIM(v)$ if the true regression function is additive across modules and if the modules are independent of one another. If our assumptions are met, the VIMs obtained by analyzing each module separately are asymptotically the same as those that would have been obtained if VIMs were obtained by analyzing all features at once.

Therefore, the selection step of fuzzy forests achieves two goals. First of all, it reduces the number of features that have to be analyzed at one time. Second, the finite sample bias caused by correlated features is alleviated. In (Nicodemus and Malley 2009), it is observed that insignificant features that are correlated with a significant feature are more likely to be chosen at the root of tree than uncorrelated significant features. The high importance of these insignificant correlated features comes at the cost of the significant uncorrelated features. When we analyze each module separately, features in different groups no longer competing against one another.

These observations suggest that if we can assume strict additivity and independence of the modules, then obtaining VIMs from each module separately should suffice. However, if these assumptions are not met, then the VIMs obtained by analyzing each module separately are, in general, different than the VIMs obtained by fitting a single random forest.

Fuzzy forests relaxes these overly restrictive assumptions. Fuzzy forests allows for interactions between features that were found to be important within modules. In biological applications,

modules might represent different biological components, or demographic information about the subjects. Thus, it is reasonable to allow these systems to interact with one another. Fuzzy forests is implicitly making the assumption that the features that have high VIMs within modules are also most likely to be the features involved in interactions across modules.

It is important to note that RFE-RF is applied at the final selection step. When features from separate modules are combined, the potential for bias due to correlation between features is re-introduced. The iterative random forests, in practice, manage to eliminate unimportant predictors which are correlated with important ones. While the ranking of the VIMs obtained from fuzzy forests are less biased than those derived from random forests, the estimated VIMs may still be biased and thus we do not recommend interpreting the estimate of the VIMs too closely.

3. The fuzzyforest package

The package **fuzzyforest** has two functions for fitting fuzzy forests. The first is `wff`, the second is `ff`. The function `wff` automatically carries out a WGCNA analysis on the features. Then it uses these newly derived modules as input to fuzzy forests. The WGCNA analysis is carried out via the `blockwiseModules` function, from the package **WGCNA**.

The second function `ff` assumes that the features have already been partitioned into separate modules either via a previous network analysis such as WGCNA, or a priori knowledge. The function `ff` is able to carry out the fuzzy forests algorithm using the output of **WGCNA**.

A number of tuning parameters must be specified before fuzzy forests can be run. These tuning parameters are organized into separate control objects. Tuning parameters related to WGCNA are specified with an `S3` object of type `WGCNA_control`. Similarly, tuning parameters related to the screening step and the selection step are specified through objects of type `screen_control` and `select_control`.

We demonstrate the workings of **fuzzyforest** with an analysis of a data set of gene expression in liver tissue in female mice. The data set can be found in the tutorial website for WGCNA: http://labs.genetics.ucla.edu/horvath/CoexpressionNetwork/Rpackages/WGCNA/Tutorials/. The number of mice is 131 and the number of genes is 3,600. We examine how the expression of these genes correlates with the weight(g) of the mice. In the following code, the data set is called `Liver_Expr`.

```
> weight <- Liver_Expr[, 1]
> expression_levels <- Liver_Expr[, -1]
```

We first use **WGCNA** to select the power that leads to a network with approximately scale-free topology. We set $\beta = 6$ (β is equivalent to `power` in the code below) and set other tuning parameters for WGCNA in the following call. Note that the resulting number of modules can be sensitive to minModuleSize.

```
> WGCNA_params <- WGCNA_control(power = 6, TOMType = "unsigned", minModuleSize = 30,
+                               numericLabels = TRUE, pamRespectsDendro = FALSE)
```

Then we set tuning parameters for the selection step and the screening step:

```
> mtry_factor <- 1; drop_fraction <- .25; number_selected <- 10
> keep_fraction <- .05; min_ntree <- 5000; ntree_factor <- 5
> final_ntree <- 5000;
> screen_params <- screen_control(drop_fraction = drop_fraction,
+                                 keep_fraction = keep_fraction,
+                                 min_ntree = min_ntree, mtry_factor = mtry_factor,
+                                 ntree_factor = ntree_factor)
> select_params <- select_control(drop_fraction = drop_fraction,
+                                 number_selected = number_selected,
+                                 min_ntree = min_ntree, mtry_factor = mtry_factor,
+                                 ntree_factor = ntree_factor)
>
```

Finally, we use `wff` to fit fuzzy forests to the data set.

```
> wff_fit <- wff(expression_levels, weight, WGCNA_params=WGCNA_params,
+               screen_params = screen_params,
+               select_params = select_params,
+               final_ntree = final_ntree,
+               num_processors = 4)
```

The function `wff` returns an object of type `fuzzy_forest`. Objects of type `fuzzy_forest` have the usual generic methods. The function `print` returns the list of selected features. The function `predict(fuzzy_forest, new_data)` takes in a `data.frame` or `matrix` and produces predictions based on the selected features.

```
> print(wff_fit)
```

	feature_name	variable_importance	module_membership
1	MMT00026944	1.0712	6
2	MMT00019254	0.8624	6
3	MMT00067823	0.8081	10
4	MMT00065159	0.6716	10
5	MMT00030931	0.5934	7
6	MMT00078732	0.5416	6
7	MMT00078851	0.5270	6
8	MMT00002575	0.5261	7
9	MMT00006001	0.5231	7
10	MMT00021649	0.5185	1

Before the analysis is run, the user selects the desired number of important features as the end output of fuzzy forests. The number of features selected can be thought of as a tuning parameter. The predictive accuracy of a test set can then be used to determine the optimal number of features to select.

As it is often useful to ascertain which modules are contributors to the signal of the outcome, we create a visual representation of all modules and the distribution of important features across the modules. The function `modplot` yields a visual display of which modules are

```
> modplot(wff_fit)
```

Figure 1: `modplot` demonstrates the relative importance of modules

important. The grey bars represent what percentage of the total p features fall into a particular module. The blue bars represent the percentage of selected features that fall into each module. Applying the function `modplot` to the object `wff_fit` above, we obtain the graph in Figure 1.

A variable importance plot can be obtained using the function `varImpPlot` from the package **randomForest**. Note that `final_rf` is the final random forest fit using the selected features.

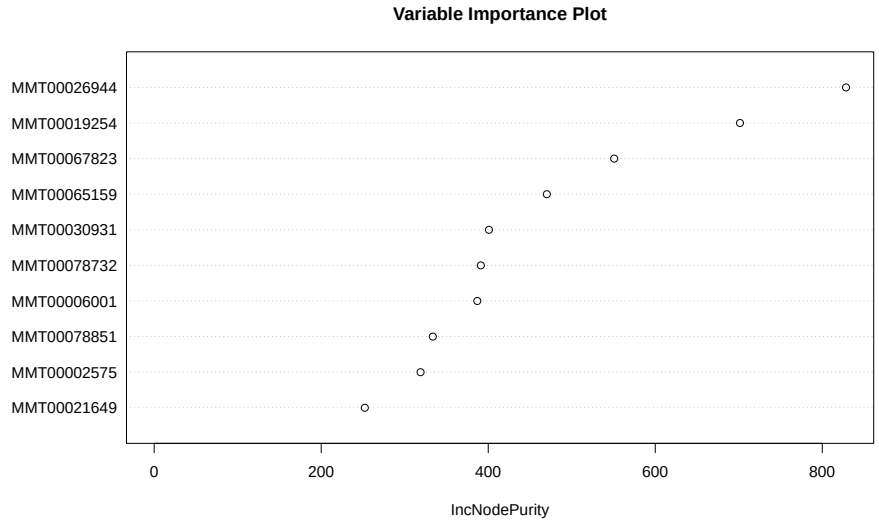
4. Simulations

In this section, we demonstrate the performance of fuzzy forests in a number of simulation scenarios. These simulations are designed to compare fuzzy forests to random forests when the features are correlated. We also demonstrate the effects of user specified tuning parameters $mtry\_factor$ and $keep\_fraction$.

In all simulations, the sample size is set to 100. In the first scenario, $p = 100$, $Y_i = X_i'\gamma + \epsilon_i$, and X_i is generated from a multivariate normal distribution, with mean 0 and variance 1. The error terms, ϵ_i are normal with mean 0 and standard deviation 0.5. Each feature has mean 0 and variance 1. Let $X^{(1)}$ through $X^{(75)}$ be correlated and let the last 25 features be independent. Among the group of correlated features, $\{X^{(1)}, \ldots, X^{(25)}\}$, $\{X^{(26)}, \ldots, X^{(50)}\}$, and $\{X^{(51)}, \ldots, X^{(75)}\}$ constitute 3 distinct modules each containing 25 features. The correlation between features within the same module is 0.8. The correlation between features in different modules is 0. The features in the final module, $\{X^{(76)}, \ldots, X^{(100)}\}$, are independent of one another and independent of features in the previously mentioned modules.

To evaluate the results of fuzzy forests and random forests, we compute the proportion of times the non-zero features were selected over 100 simulation runs. For random forests, the features with permutation VIM in the top 10 were selected. Among the correlated features

```
> varImpPlot(wff_fit$final_rf, type = 2, main="Variable Importance Plot")
```



we have $\gamma_1 = \gamma_2 = 5$ and $\gamma_3 = 2$. Among the group of independent features, $\gamma_{76} = \gamma_{77} = 5$ and $\gamma_{78} = 2$. All other elements of γ were set to 0. The results are displayed in Figure 2.

In the second scenario, we have the same setup as before except we increase p to 1000 while leaving n at 100. The group of correlated features now contains 900 features, grouped into the following modules:
$\{X^{(1)}, \ldots, X^{(100)}\}, \ldots, \{X^{(801)}, \ldots, X^{(900)}\}$. Again, the correlation between features in the same module is 0.8. The correlation of features from different modules is 0. The remaining module, $\{X^{(901)}, \ldots, X^{(1000)}\}$, consists of independent features. Once again, $\gamma_1 = \gamma_2 = 5$ and $\gamma_3 = 2$. The first 3 independent features are also non-zero: $\gamma_{901} = \gamma_{902} = 5$ and $\gamma_{903} = 2$. As seen in Figure 3, when $p = 1,000$, random forests largely ignore the independent features.

5. Application

We demonstrate a typical analysis by using **fuzzyforest** to discover immunologic profiles that predict if an HIV infected patient will have a suboptimal immune response to antiretroviral therapy (ART). The subjects in the study were enrolled in either the Options Project or the SCOPE project, both longitudinal cohorts based at the University of California, San Francisco (UCSF). In this particular analysis, we wish to identify novel immunologic signatures that are predictive of whether a patient will be an immunologic responder or a controller, which we define as being able to achieve undetectable levels of the virus (< 50 copies/ml) without ART. Similarly, an immunologic responder is an aviremic patient with undetectable levels of the virus and CD4+ T cell counts above 350 cells/mm3.

In this dataset there were 125 immunologic responders, 92 controllers, and 313 features. The features are derived flow cytometry, measuring T cell maturation, activation, dysfunction, senescence, antigen-specificity and proliferation as well as patient demographics and clinical characteristics. Flow cytometry measures up to 14 different markers on a cell. The data are labeled “n” for negative and “p” for positive indicating the presence or absence of the

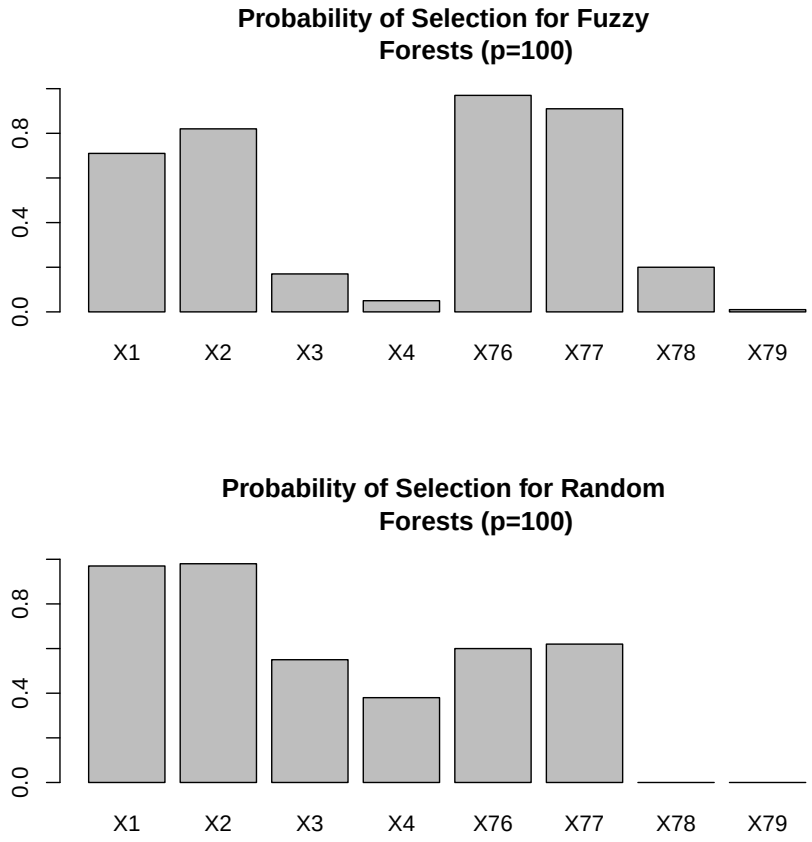


Figure 2: Fuzzy forests compared to random forests with $p = 100$. Each bar represents the probability of each feature being selected. X_1 through X_4 are correlated features and X_{76} through X_{79} are independent features. X_1 through X_3 and X_{76} through X_{78} are non-zero in the true model.

marker on the cell. This yields up to 2^{14} possible binary combinations of markers. Due to cost, not all permutations were done. Due to the nature of flow-cytometery, many features are subsets of other features and hence highly correlated. Mean florescence intensity was also measured using flow cytometry, but in this case the measure is continuous. We were also given information on clinical and demographic characteristics of the subjects such as gender, age, ART regimen, and route of transmission. Because we have many highly correlated features we used WGCNA to identify modules within the feature space. We used the scale-free topology criterion to determine the power of the adjacency function and set $\beta = 8$ based on the results of Figure 4. We found 11 modules. The largest module was the "grey" module, the features that are independent of the other modules, with 140 features. It is commonly the case that the grey module is larger than the other modules. The smallest module was of size 10.

Here is description of modplot.

We used the resulting modules memberships as input to the function `ff`. Because of the small size of the modules we set `keep_fraction` to 0.25. We tested multiple values of for

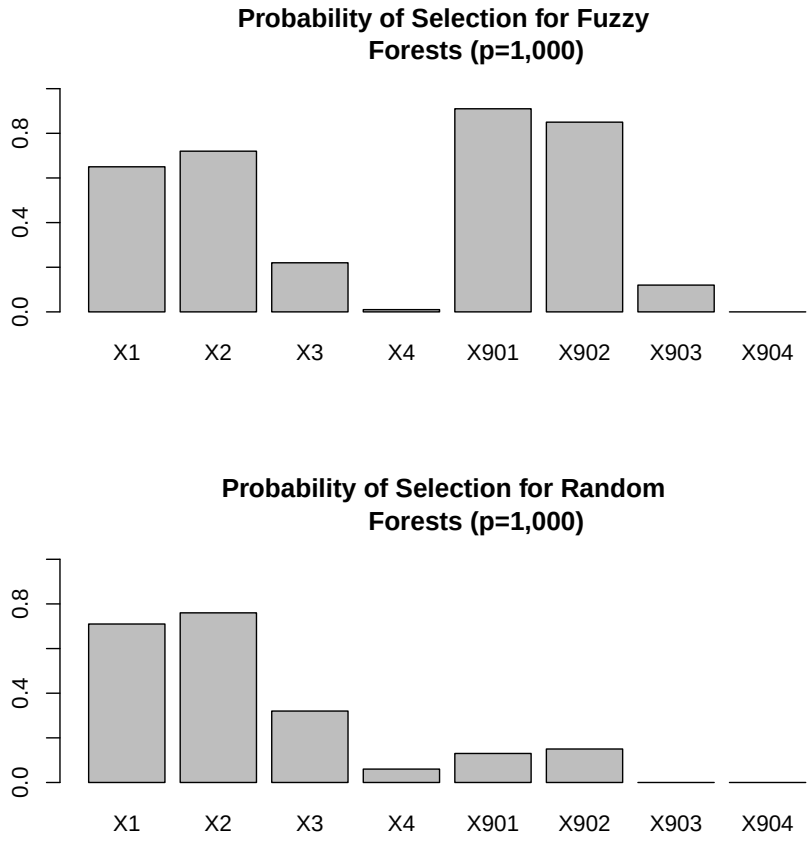


Figure 3: Fuzzy forests compared to random forests with $p = 1,000$. Each bar represents the probability of each feature being selected. X_1 through X_4 are correlated features and X_{901} through X_{904} are independent features. X_1 through X_3 and X_{901} through X_{903} are non-zero in the true model.

`number_selected`. The ranking of features was robust to settings of this parameter. We display the results when selecting 10 features.

The strongest predictors of virologic control without ART were HIV GAG-specific response and higher levels of immune activation. This replicates findings from earlier work Hunt *et al.* (2008). Interestingly immune responders had higher percentages of CD4 positive cells that express both CCR5 and CD38, but not HLA-DR. Immune responders also had higher levels of CD31 positive naive cells.

6. Discussion

In this paper we have presented the fuzzy forests algorithm as an extension of random forests that can provide less biased feature selection in the presence of correlation between features especially when $p >> n$. Under these conditions fuzzy forests is expected to outperform random forests. We found that, as expected, random forest variable importance measures

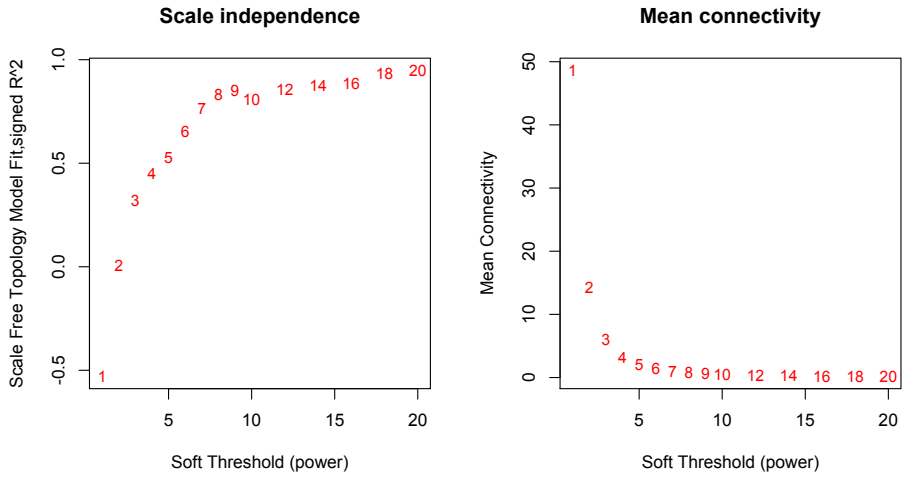


Figure 4: The following plot shows that 8 is the smallest power such that the scale free topology criterion is approximately met.

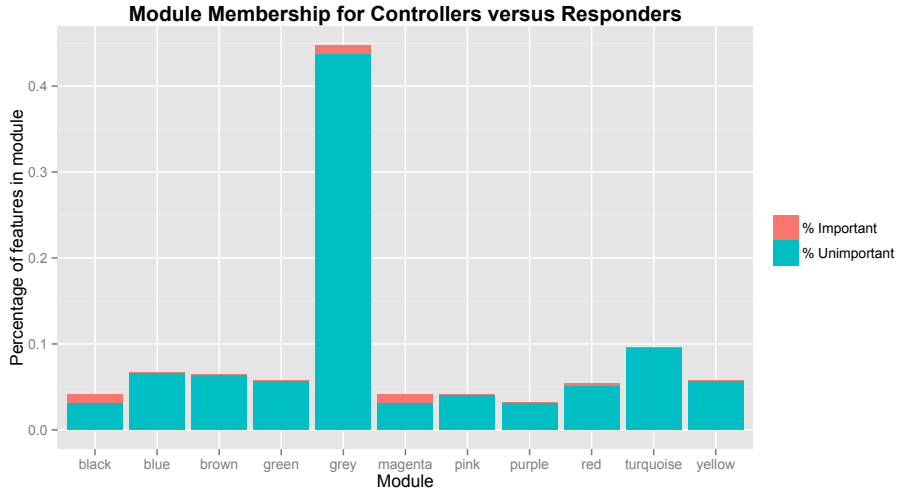


Figure 5: The height of the bars represents the proportion features in each module. The proportion of each bar colored in red represents the proportion of features that are selected as important within each module.

were biased towards correlated features. Indeed when $p = 1,000$ while $n = 100$, random forests essentially ignored the independent variables that were a priori important in the true model while fuzzy forests found them. The fuzzy forests algorithm is useful for screening large numbers of parameters or when it is desirable to find the top number of features contributing to the signal. Of course, this reduced list can be expected to have slightly better predictive performance, although this was not the original intent of the algorithm.

We introduced an implementation of fuzzy forests in the **fuzzyforest** package. The **fuzzyforest** package has two functions for fitting the fuzzy forests algorithm. The first implementation,

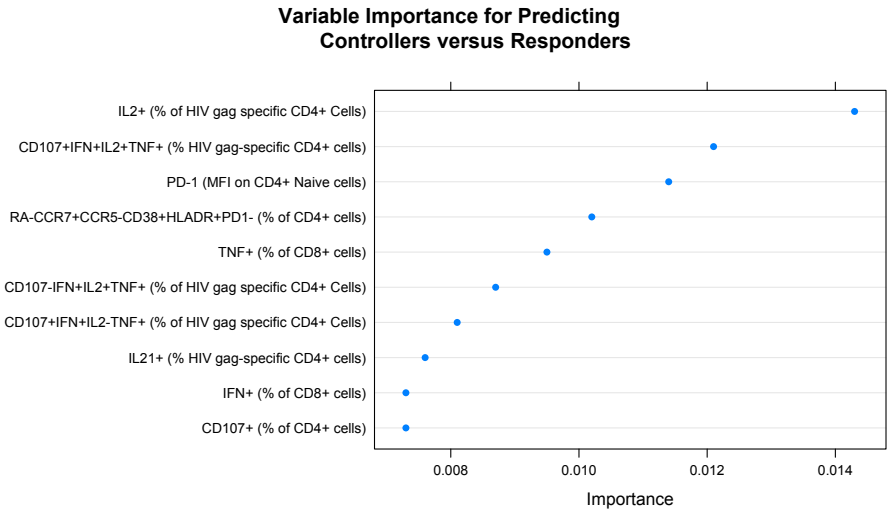


Figure 6: The following plot displays the importance of the top 10 selected features after fitting a fuzzy forest. The variables are ranked from top to bottom.

`wff` automatically carries out WGCNA to partition the features into separate modules. These modules are then used by the fuzzy forests algorithm for feature selection. The second implementation, `ff` lets the user determine how features should be partitioned before the fuzzy forests algorithm is used for feature selection.

We then used fuzzy forests to investigate how immunologic profiles determine a patient's response to HIV both with and without ART. We used the scale free topology to determine the power β of the adjacency function. Then we used the `ff` function for feature selection. The set of important features was stable with respect to $mtry\_factor$ and other tuning parameters. The set of features found by fuzzy forests is biologically plausible and in part confirms findings from in vivo and other clinical studies, suggesting that fuzzy forests found the true underlying signal in the data. It is expected that fuzzy forests will be useful in a wide variety of application from gene studies, to flow cytometry to other studies where the data has high correlation and many potential predictors.

Acknowledgements

This work was partially funded by NSF IIS 1251151 and AMFAR 8721SC.

References

Bondell HD, Reich BJ (2008). "Simultaneous regression shrinkage, variable selection, and supervised clustering of predictors with OSCAR." *Biometrics*, **64**(1), 115–123.

Breiman L (2001). "Random Forests." *Machine learning*, **45**(1), 5–32.

Díaz-Uriarte R, De Andres SA (2006). "Gene Selection and Classification of Microarray Data Using Random Forest." *BMC bioinformatics*, **7**(1), 3.

Dua S, Acharya UR, Dua P (2014). *Machine learning in healthcare informatics*. Springer.

Gregorutti B, Michel B, Saint-Pierre P (2013). "Correlation and Variable Importance in Random Forests." *arXiv preprint arXiv:1310.5726*.

Horvath S (2011). *Weighted Network Analysis: Applications in Genomics and Systems Biology*. Springer.

Hunt PW, Brenchley J, Sinclair E, McCune JM, Roland M, Shafer KP, Hsue P, Emu B, Krone M, Lampiris H, *et al.* (2008). "Relationship Between T Cell Activation and CD4+ T Cell Count in HIV-Seropositive Individuals with Undetectable Plasma HIV RNA Levels in the Absence of Therapy." *Journal of Infectious Diseases*, **197**(1), 126–133.

Langfelder P, Horvath S (2008). "WGCNA: an R package for Weighted Correlation Network Analysis." *BMC bioinformatics*, **9**(1), 559.

Nicodemus KK, Malley JD (2009). "Predictor correlation impacts machine learning algorithms: implications for genomic studies." *Bioinformatics*, **25**(15), 1884–1890.

Raskutti G, Wainwright MJ, Yu B (2010). "Restricted Eigenvalue Properties for Correlated Gaussian Designs." *The Journal of Machine Learning Research*, **11**, 2241–2259.

Strobl C, Boulesteix AL, Kneib T, Augustin T, Zeileis A (2008). "Conditional Variable Importance for Random Forests." *BMC bioinformatics*, **9**(1), 307.

Strobl C, Boulesteix AL, Zeileis A, Hothorn T (2007). "Bias in Random Forest Variable Importance Measures: Illustrations, Sources and a Solution." *BMC bioinformatics*, **8**(1), 25.

Zhang B, Horvath S (2005). "A General Framework for Weighted Gene Co-Expression Network Analysis." *Statistical applications in genetics and molecular biology*, **4**(1).

Zhu R, Zeng D, Kosorok MR (2012). "Reinforcement Learning Trees."

Affiliation:

Daniel Conn
Department of Biostatistics
UCLA School of Public Health
Los Angeles, United States of America
E-mail: djconn17@gmail.com