Journal of Statistical Software

October 2019, Volume 91, Issue 9. doi: 10.18637/jss.v091.i09

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

Daniel Conn
University of California, Los Angeles

Tuck Ngun
University of California, Los Angeles

Gang Li
University of California, Los Angeles

Christina M. Ramirez
University of California, Los Angeles

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 the number of features, p, is much larger than the sample size, n ($p \gg n$). 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 the package **WGCNA** (weighted gene coexpression network analysis, alternatively known as weighted correlation network analysis) 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, $p \gg n$, 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 features (p) to be much larger than the number of observations (n), the so-called $p \gg n$ 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 least absolute shrinkage and selection operator (LASSO, Tibshirani 1996; Friedman, Hastie, and Tibshirani 2010; Hastie, Tibshirani, and Wainwright 2015) or smoothly clipped absolute deviation (SACD, Fan and Li 2001; Breheny and Huang 2011) may detect important features in the presence of correlation (Raskutti, Wainwright, and Yu 2010), this comes at the cost of making parametric assumptions that may not hold in practice.

Random forests are a popular ensemble machine learning algorithm. Random forests are nonparametric, nonlinear, embarrassingly parallelizable, easy to implement, and have been described as one of the best "off-the-shelf" classifiers (Dua, Acharya, and Dua 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, Boulesteix, Zeileis, and Hothorn 2007; Strobl, Boulesteix, Kneib, Augustin, and Zeileis 2008; Nicodemus and Malley 2009).

Fuzzy forests handle correlated features by taking a piecewise approach. We first estimate the correlation 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. We then use recursive feature elimination random forests (RFE-RF, Díaz-Uriarte and Alvarez de Andrés 2006) to select the most important features from each module. The surviving features from each module are combined and one final RFE-RF is then applied, selecting and ranking the most important ones. The fact that fuzzy forests carry out separate feature selection algorithms on distinct groups of correlated covariates distinguishes it from other commonly used random forest based feature selection methods. We believe that fuzzy forests will be useful to a wide variety of researchers including those in biology, medicine, psychology, social sciences, and any application in which there is high dimensional data with correlation.

The general fuzzy forests algorithm allows for the use of a variety of methods for partitioning the features into distinct clusters. The **fuzzyforest** (Conn, Ngun, and Ramirez 2019) package allows the analyst to input their own clustering of the features. Commonly, such a partition of the features would be derived by considering the correlation matrix of the features.

The particular implementation of the fuzzy forests algorithm given in the R package **fuzzyforest** also gives the analyst the option of utilizing the functionality of weighted gene coexpression network analysis via the package **WGCNA** (Langfelder and Horvath 2008, 2012) to partition covariates into distinct clusters. WGCNA is a rigorous framework for detecting correlation networks (Zhang and Horvath 2005). Although WGCNA has been used primarily in genetics, it has also been applied successfully in contexts such as brain imaging and cancer biology (Langfelder and Horvath 2008).

The conditional variable importance measures introduced in Strobl *et al.* (2008) have also been

proposed as a means for reducing the bias in random forest VIMs. However, the calculation of conditional variable importance measures is computationally intensive. In this article, we compare feature selection from random forests, conditional inference forests, and fuzzy forests, using packages **randomForest** (Liaw and Wiener 2002), **party** (Hothorn, Bühlmann, Dudoit, Molinaro, and Van der Laan 2006; Strobl *et al.* 2007, 2008), and **fuzzyforest**, respectively. We find that fuzzy forests offer a computationally feasible alternative to conditional inference forests for feature selection in the presence of highly correlated features.

2. Variable importance measures and fuzzy forests

2.1. Motivation for variable importance measures

In this section we introduce basic notation and discuss VIMs. The VIMs that we discuss in this section describe important aspects of the true regression function and are well-defined outside of the context of random forests. 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, with vth element $X^{(v)}$, and Y is a scalar outcome. Let $X_i^{(v)}$ denote the value of the vth feature for the ith subject and let $X_i = (X_i^{(1)}, \ldots, X_i^{(p)})^\top$ be the feature vector for the ith subject. Finally, the distribution of X and the marginal distribution of $X^{(v)}$ are denoted as G_X and $G_{X^{(v)}}$, respectively.

In the case of 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 $f(X)$. We assume that $Y|X$ has distribution equal to that of $f(X) + \epsilon$, where Y is continuous and the ϵ are independent of X and iid with variance σ^2. In the case of regression, a prediction for a new observation X_{new} would be obtained by evaluating the conditional mean at X_{new}: $f(X_{new})$.

For binary classification, we are again interested in modeling the conditional mean of Y given a feature vector X, however, Y is restricted to take the value 0 or 1. Thus, $Y|X$ is a Bernoulli trial with mean $E[Y|X = x] = P(Y = 1|X = x)$. 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. Random forests are also able to handle the case of multi-class classification (Breiman 2001).

For both classification and regression, we say that feature $X^{(v)}$ is unimportant if $E[Y|X]$ does not depend on $X^{(v)}$. 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 a 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 $f_\gamma(X_i) = \gamma_0 + \sum_{v=1}^{p} \gamma_v X_i^{(v)}$, 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 nonlinear and nonparametric. Therefore, the resulting random forest VIMs, defined below, naturally take interactions and nonlinear structure into account.

We would like to end this section with a discussion of the general goals of feature selection and how they relate to estimation of VIMs. The ultimate objective of fuzzy forests is to select a small subset of features such that the selected features will have relatively high VIM in comparison with the rest of the features. Another potential goal of feature selection is to select a subset of features with the ultimate goal of predicting outcomes for new observations. In this latter case, feature selection might be advisable as a means of improving predictive capabilities (some predictive algorithms may be adversely effected by the presence of unimportant features). Using feature selection with the goal of prediction may also be useful if it is advantageous to reduce the number of measurements taken on each individual.

In the presence of correlation, feature selection methods such as fuzzy forests that are designed to select features with the highest VIMs may yield different results than feature selection methods designed to optimize predictive accuracy. For example, suppose two features are highly correlated and only one feature is important while the other is not. If the goal is maximizing predictive accuracy, either feature may be selected without adversely effecting predictive ability. The two features effectively serve as proxies for one another.

2.2. An introduction to random forests

The random forests algorithm is a popular ensemble method that has been applied in the settings of both classification and regression (Breiman 2001). 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." Roughly one third of subjects will be out of bag for each tree. These out of bag subjects play the important role of serving as a validation 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 tree, by itself, may be highly unstable, leading to highly variable 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, derived from 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*. If $mtry = p$, then random forests are equivalent to bagging. High values of *mtry* tend to lead to just a few important features getting selected at the majority of nodes. Lower values of *mtry* allow more features to play a

role in the estimation of $f(X)$. In the case of regression, a common default value of *mtry* is $\lfloor p/3 \rfloor$ and, in the case of classification, $\sqrt{p}$ is a common choice (Liaw and Wiener 2002).

Multiple random forest VIMs have been developed. In this article we will exclusively focus on unscaled random forest permutation VIMs. Random forest permutation VIMs are obtained by testing how predictive accuracy suffers when the values of an individual feature are randomly permuted. For example, suppose a particular feature is important in determining the value of the outcome. Randomly permuting the values of this feature destroys its relationship with the outcome. Because the connection between this particular feature and the outcome has been obscured, there should be a subsequent decrease in predictive accuracy when predictions are made using this permuted data. If there was no relationship to begin with, the predictive accuracy obtained using the permuted data should be comparable to the predictive accuracy obtained using the original, unpermuted, data. The random forest permutation VIM measures the average decline in predictive performance for each feature across multiple trees.

We now describe the calculation of the random forest permutation VIM for the vth feature. Let $OOB_k \subset \{1, \ldots, n\}$ be the indices for the out of bag sample from the kth tree and let $|OOB_k|$ be the number of out of bag samples. Let $\pi_k = (\pi_{k1}, \ldots, \pi_{kn})$ be a random permutation of OOB_k and let $\tilde{X}_i = (X_i^{(1)}, \ldots, X_{\pi_{ki}}^{(v)}, \ldots, X_i^{(p)})^\top$ 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 random forest permutation VIM for the vth feature is defined as

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

We note that a number of other VIMs are in common use. The **randomForest** package implements two type of VIMs. **randomForest** implements the permutation based VIM discussed above. It also implements a VIM based on the mean decrease in "impurity" in the child nodes after splitting a node on a particular feature. The measure of impurity will depends on whether classification or regression trees are being used. For example, in the case of regression, the within-node variance is a measure of impurity. For classification, the Gini-index is the default measure of node impurity.

In the package **party**, an additional VIM, called the conditional VIM is implemented. The conditional VIM, developed in Strobl *et al.* (2008), has been shown to reduce the bias in random forest VIMs, however, calculation of the conditional VIM is computationally quite expensive, particularly when the sample size is large.

We summarize a number of VIMs and feature selection methods in Table 1 below. There is a distinction between VIMs and feature selection methods. VIMs alone only give a ranking of the features in terms of their importance. Once VIMs have been calculated, the resulting ranking can then be used for feature selection. Thus, VIMs may play the central role in a feature selection procedure. For example, calculating random forest VIMs and keeping the VIMs that rank in the top 5% defines a feature selection procedure. The fuzzy forests algorithm is a more complex feature selection procedure that relies on the calculation of VIMs.

Method	Type	Nonparametric	R package
Permutation importance	VIM	T	**randomForest/party**
Mean decrease in node impurity	VIM	T	**randomForest**
Conditional VIM	VIM	T	**party**
Fuzzy forests	FS	T	**fuzzyforest**
LASSO	FS and VIM	F	**glmnet**
SCAD	FS and VIM	F	**ncvreg**

Table 1: Popular VIM and feature selection (FS) methods.

Nicodemus, Malley, Strobl, and Ziegler (2010) present a clear discussion of the nature and source of bias in random forest permutation VIMs. In this article, Nicodemus *et al.* (2010) conduct a simulation study in which the true model is linear with a group of positively correlated important features and a group of independent important features. They find in this simulation study that permutation VIMs favor the group of correlated features as the correlated features have higher marginal correlation with the outcome compared to the independent features. They find this bias to be comparable to the bias observed in the context of linear models. Just as univariate regressions of the outcome on each feature would favor the correlated important features, the permutation VIMs favor the correlated important features. However, the permutation VIMs' bias, while present, is somewhat smaller than that of univariate regressions.

It is worthwhile noting that the correlation of features alone does not guarantee heavily biased permutation VIMs. For example, as demonstrated in Nicodemus *et al.* (2010), in a null model, a model in which none of the features are important for the outcome, the resulting permutation VIMs are not particularly biased. Correlation of features will induce bias in the VIMs if the correlation structure induces marginal correlations that do not reflect the importance of the features.

2.3. A brief review of WGCNA

In genetics, 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 biological applications. The R package **WGCNA** is a robust and well-documented implementation of the WGCNA framework (R Core Team 2019; Langfelder and Horvath 2008) that was originally designed to detect correlation networks in the context of genetics. Despite the acronym, WGCNA has been used extensively outside of the context of gene expression data. For example, it has seen use in the analysis of fMRI data (Mumford, Horvath, Oldham, Langfelder, Geschwind, and Poldrack 2010). We believe that WGCNA has fairly wide applicability as, at its core, it relies on an application of hierarchical clustering methods to functions of the correlation matrix.

We expect that researchers already familiar with the **WGCNA** package will easily adopt the fuzzy forests algorithm and we expect that newcomers to **WGCNA** will be able to make good use of **WGCNA**'s fine documentation and tutorials. WGCNA takes in the matrix of features and uses the correlation structure to partition the features into distinct groups such that the correlation between features in the same group is large and the correlation between features in separate groups is small. In the context of WGCNA, these groups of features are called

modules. WGCNA constructs a network of features, each feature representing one node, via the correlation matrix of features. It determines modules based off of this network.

Formally, the user first specifies a similarity matrix with (u, v)th entry $s_{uv} = S(X^{(u)}, X^{(v)})$ for features u and v. The function $S(X^{(u)}, X^{(v)})$ is called the similarity function and often takes values between 0 and 1. Common similarity functions include $|\widehat{Corr}(X^{(u)}, X^{(v)})|$ and $(1 + \widehat{Corr}(X^{(u)}, X^{(v)}))/2$ (Zhang and Horvath 2005), where $\widehat{Corr}(X^{(u)}, X^{(v)}))$ is the sample correlation between features u and v.

This similarity matrix is then transformed into an adjacency matrix $A = [a_{uv}]$ via an adjacency function $a_{uv} = \alpha(s_{uv})$. The adjacency function determines how similarities translate into properties of the network. The 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 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 a 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. 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 $q_{uv} = \sum_{r=1}^{p} a_{ur} a_{rv}$ and $c_u = \sum_{r=1}^{p} a_{ur}$ is the connectivity of the uth feature (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 the topological overlap matrix rather than the adjacency matrix 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 β to select. A network is said to have generalized scale-free topology if $r(c_u) \propto c_u^{\beta}$, or, equivalently, $\log_{10}(r(c_u)) \propto \log_{10}(c_u)$ (Zhang and Horvath 2005), where $r(c_u)$ is the frequency function for the connectivity and β is a non-negative real number. If the scale-free topology criterion is suspected to hold, one should select a value of β such that the R^2 between $\log_{10}(r(c_u))$ and $\log_{10}(c_u)$ is high.

2.4. The fuzzy forests algorithm

The fuzzy forests algorithm is an extension of random forests designed to obtain less biased feature selection in the presence of correlated features. In this section, we describe the algorithm. First we give a summary of the procedure.

In the first step of fuzzy forests, the features are partitioned into distinct groups or modules, such that the correlation of features within modules is high and the correlation of features between modules is low. Our package, **fuzzyforest**, facilitates the use of **WGCNA** to determine the modules although it is possible to use alternative methods to partition the features. Once features have been subdivided into distinct modules, fuzzy forests eliminates features in two steps: a screening step and a selection step. In the screening step, RFE-RF is used on each

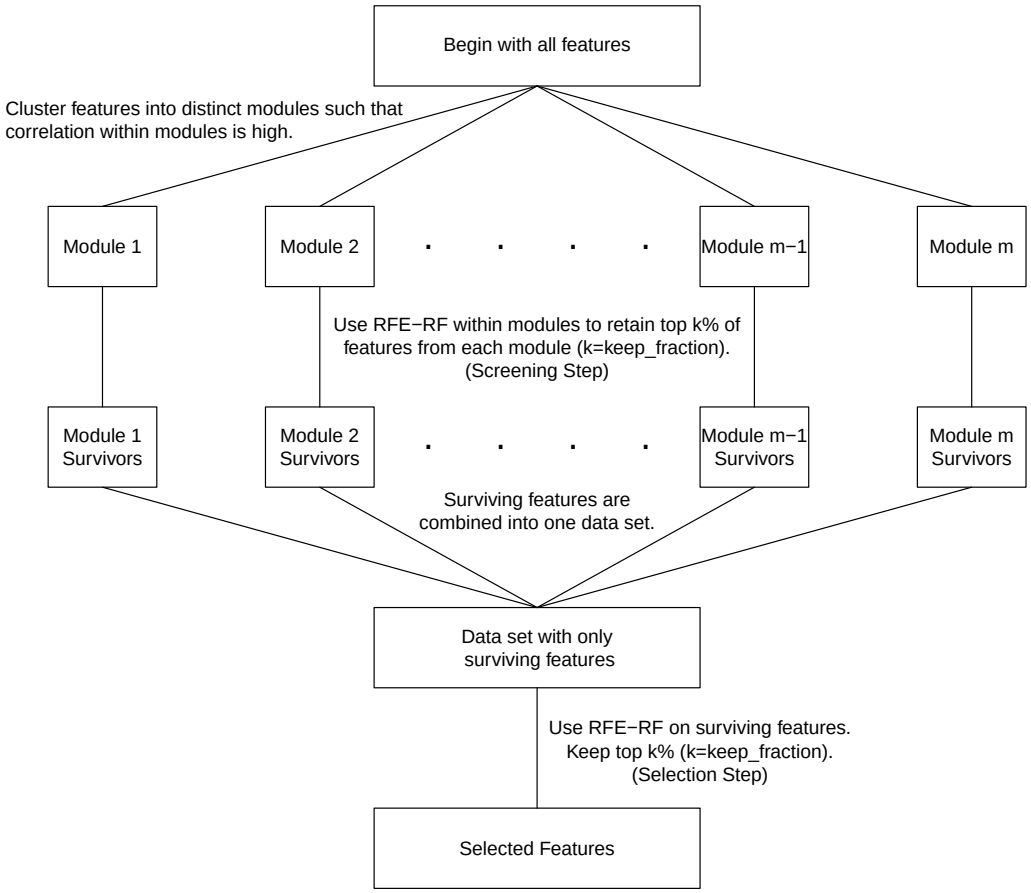

Figure 1: Flow chart of fuzzy forests algorithm.

module to eliminate the least important features within each module. In the selection step, a final RFE-RF is used on the surviving features.

A detailed explication of RFE-RF is given below. RFE-RF sequentially eliminates features with the lowest VIMs until a pre-specified percentage of features remain. By sequentially eliminating the least important features, RFE-RF is able to better focus on determining which features are the most important.

The screening step of fuzzy forests achieves two goals. First, 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 unimportant features that are correlated with an important feature are more likely to be chosen at the root of tree than uncorrelated important features. The high importance of these unimportant correlated features comes at the cost of the important uncorrelated features. When we analyze each module separately, features in different groups are no longer competing against one another.

In biological applications, modules might represent different biological components or demographic information about the subjects. By carrying out RFE-RF on the features that survived the screening step, the selection step effectively allows these systems to interact with one another. A flow chart of the fuzzy forests algorithm is given in Figure 1.

We now provide a detailed description of the screening step and RFE-RF. Denote the set of modules by $P = \{P_1, \ldots, P_m\}$. Let $p_l = |P_l|$ so that $\sum_{l=1}^{m} p_l = p$, where m is the number of modules. For each element of the partition, P_l, RFE-RF is used to screen out unimportant features.

We now describe the RFE-RF procedure in the context of screening features in a particular partition P_l. At the start of the procedure, a random forest is fit using all of the features in P_l and the least important features are then eliminated. For example, the features with VIM in the bottom 25% might be dropped. Call the reduced set of features in P_l, after this first random forest, $P_l^{(1)}$. A second random forest is then fit using only 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 what fraction of 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 fraction of the original p_l features, in each module P_l, to retain. This fraction is called the `keep_fraction`. The first time the number of features drops below $\texttt{keep\_fraction} \cdot p_l$, the RFE-RF stops and the top $\lfloor \texttt{keep\_fraction} \cdot p_l \rfloor$ features are selected. More precisely, for the first iteration t such that $p_l^{(t)} < \texttt{keep\_fraction} \cdot p_l$, we retain the top $\lfloor \texttt{keep\_fraction} \cdot p_l \rfloor$ features from $P_l^{(t-1)}$.

For each RFE-RF, `mtry` and `ntree` must be appropriately selected. Since the number of features varies across the forests, `mtry` and `ntree` must be a function of the current number of features. Suppose we are at iteration t and are about to fit a random forest to obtain $P_l^{(t+1)} \subset P_l^{(t)}$, in the case of regression, **fuzzyforest** sets $\texttt{mtry} = \left\lfloor p_l^{(t)} \cdot \texttt{mtry\_factor}/3 \right\rfloor$. For classification, **fuzzyforest** sets $\texttt{mtry} = \left\lfloor \sqrt{p_l^{(t)} \cdot \texttt{mtry\_factor}} \right\rfloor$. In both cases, `mtry_factor` must be 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 takes longer to fit. The package **fuzzyforest** sets $\texttt{ntree} = \max(\texttt{min\_ntree}, \left\lfloor p_l^{(t)} \cdot \texttt{ntree\_factor} \right\rfloor)$, where `min_ntree` is a minimal number of trees grown for each forest and `ntree_factor` allows the number of trees to increase with the number of features.

The final step consists of one last RFE-RF to allow for interactions between features in different modules. Note that a separate choice of `drop_fraction`, `mtry_factor`, `min_tree`, and `ntree_factor` may be used for the final selection step. The user specifies how many features to keep in the final selection step. If certain features are, a priori, known to be important (perhaps demographic characteristics), **fuzzyforest** allows the user to let these features skip the initial round of screening.

Finally, we compare the RFE-RF procedure described above to the RFE-RF procedure described in Díaz-Uriarte and Alvarez de Andrés (2006) and implemented in the package **varSelRF** (Díaz-Uriarte 2007). In the procedure presented in Díaz-Uriarte and Alvarez de Andrés (2006), Díaz-Uriarte and Alvarez de Andrés (2006) prefer that the VIMs not be recalculated at each iteration, with the intent of preventing overfitting. In the RFE-RF procedure

described above, permutation VIMs are calculated at each iteration. This is because we wanted to allow the ranking of VIMs to change as unimportant features are dropped. The ultimate focus of fuzzy forests is to select features with the highest ranking VIMs. We do note that **varSelRF** does allow for the option of recalculating VIMs at each iteration and an RFE-RF procedure in which VIMs were calculated at each iteration was proposed by Jiang *et al.* (2004). Classification as opposed to regression also appears to be the primary focus of the RFE-RF procedure in **varSelRF**.

2.5. A justification for the fuzzy forests algorithm

The screening of features within distinct modules is motivated by the following heuristic observations concerning the theoretical properties of VIMs. As noted previously, correlation between features can cause bias because the correlation structure can induce high marginal correlation between features and the outcome that do not reflect the importance of the features. This is particularly problematic when important features are correlated with one another. In this case, the marginal correlation between these features and the outcome will be greatly amplified by the correlation, causing the permutation VIMs to ignore or underrate the independent and important features.

Intuitively, dividing features into distinct, correlated modules and carrying out feature selection within each module provides an advantage because the correlation structure within a module is relatively uniform. Although the correlation within a module is high, the uniform nature of the correlation structure is not expected to lead to particularly misleading VIMs. Importantly, feature selection on the independent features is unaffected by feature selection on the correlated features. In the following discussion, we formally define the parameter being estimated by permutation VIMs and we discuss conditions under which the VIMs calculated within a module are equivalent to VIMs calculated using the full set of features.

The estimation of VIMs is formally investigated by Van der Laan (2006). The random forest VIM is discussed in Gregorutti, Michel, and Saint-Pierre (2017) and Zhu, Zeng, and Kosorok (2015). Intuitively, the random forest VIM of the vth feature measures how much $f(X_i)$ changes when the vth entry of X_i, $X_i^{(v)}$, is replaced by an independent realization, $\tilde{X}_i^{(v)}$, generated with distribution $G_{X^{(v)}}$. Formally, 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 $G_{X^{(v)}}$. 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 $f_\gamma(X) = \gamma_0 + \sum_{v=1}^{p} \gamma_v X^{(v)}$ is a linear model, with standardized features ($Var(X_i^{(v)}) = 1$), $VIM(v) = 2\gamma_v^2$.

Let $G_{P^{(l)}}$ denote the joint distribution of the features in the module $P^{(l)}$ 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 $E[2\epsilon(f(X) - h(X^{P^{(l)}}))] = 0$ because ϵ is independent of X and has mean 0.

Assume 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 form of the regression function, $f(X)$, allows for interactions within modules but no interactions between modules. We now demonstrate that if we fit random forests using only the features in $P^{(l)}$, we are no longer estimating $E[Y|X] = f(X)$, instead we are estimating

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

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 using the full set of features.

This is seen by the following argument:

$$\begin{aligned}
E[Y|X^{P^{(l)}}] &= \text{argmin}_h E[(Y - h(X^{P^{(l)}}))^2] && (11)\\
&= \text{argmin}_h E[\{(Y - f(X)) - (h(X^{P^{(l)}}) - f(X))\}^2]. && (12)
\end{aligned}$$

This last term equals:

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

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

$$\begin{aligned}
E[(Y - f(X))(h(X^{P^{(l)}}) - f(X))] &= E[E[(Y - f(X))(h(X^{P^{(l)}}) - f(X))|X]] && (14)\\
&= E[(h(X^{P^{(l)}}) - f(X))E[(Y - f(X))|X]] && (15)\\
&= 0. && (16)
\end{aligned}$$

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 $j \neq l$. This yields Equation 10.

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

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

Here, $X_i^{l_k}$ is the kth element of partition $P^{(l)}$. As in Equation 4, $X_i^{(v)}$ and $\tilde{X}_i^{(v)}$ 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 each other. 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.

These observations suggest that if we assume strict additivity and independence of the modules, then obtaining VIMs from each module separately should suffice. However, if these assumptions are not met, the VIMs obtained by analyzing each module separately are, in general, different than the VIMs obtained by fitting a single random forest on all of the features at once. We stress that the above derivation depends on the additivity assumption and the assumption of independence of modules.

If there are interactions between features in different modules, the VIMs calculated within modules will be asymptotically biased. However, under the circumstances that the most important VIMs in each module are also the features that are most likely to be heavily involved in interactions between modules, carrying out feature selection on each module separately should still allow for the selection of important features.

The final RFE-RF, applied at the selection step serves to relax this restrictive additivity assumption, allowing for interactions between features that were found to be important within modules. However, it is important to note that when features from separate modules are combined, the potential for bias due to correlation between features is reintroduced. Thus, the estimated VIMs may still be biased and must be interpreted with caution.

While the implicit assumptions underlying fuzzy forests are strict, we point out that random forests, as well as conditional inference forests may also demonstrate bias. In the case of random forests, as discussed above, the simulation results of Strobl *et al.* (2008) and Nicodemus *et al.* (2010) suggest that random forests will be unbiased if the marginal correlations between features and the outcome largely reflect the true VIMs. We believe that this is an even more stringent assumption than the assumptions made in Section 2.5. We believe that fuzzy forests will have less biased feature selection properties than random forests because the conditions under which random forest feature selection is roughly unbiased are even more stringent than fuzzy forests. The simulations carried out below also demonstrate that feature selection using conditional permutation VIMs can also demonstrate bias.

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. For example, it may be advantageous to use hierarchical clustering directly on the correlation matrix and to cut the tree by visual inspection via calls to `hclust` and `cutree` in the **stats** package. This procedure may give the user more flexibility in which distance metric to use and in how to cluster the features. The package **pvclust** calculates p-values to assess the uncertainty in clusters of features and can be used to find a stable clustering of the

features. Another common use case for `ff` is to carry out the fuzzy forests algorithm using the output of **WGCNA**, thereby allowing more customization of options for **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 concerning gene expression in liver tissue in female mice. The data set can be found in the tutorial website for **WGCNA**: https://horvath.genetics.ucla.edu/html/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`.

```
R> weight <- Liver_Expr[, 1]
R> 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`.

```
R> WGCNA_params <- WGCNA_control(power = 6, minModuleSize = 30,
+    TOMType = "unsigned", reassignThreshold = 0, mergeCutHeight = 0.25,
+    numericLabels = TRUE, pamRespectsDendro = FALSE)
```

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

```
R> mtry_factor <- 1
R> drop_fraction <- .25
R> number_selected <- 10
R> keep_fraction <- .05
R> min_ntree <- 5000
R> ntree_factor <- 5
R> final_ntree <- 5000
R> 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.

```
R> wff_fit <- wff(expression_levels, weight, WGCNA_params = WGCNA_params,
+    screen_params = screen_params, select_params = select_params,
+    final_ntree = final_ntree, num_processors = 1)
```

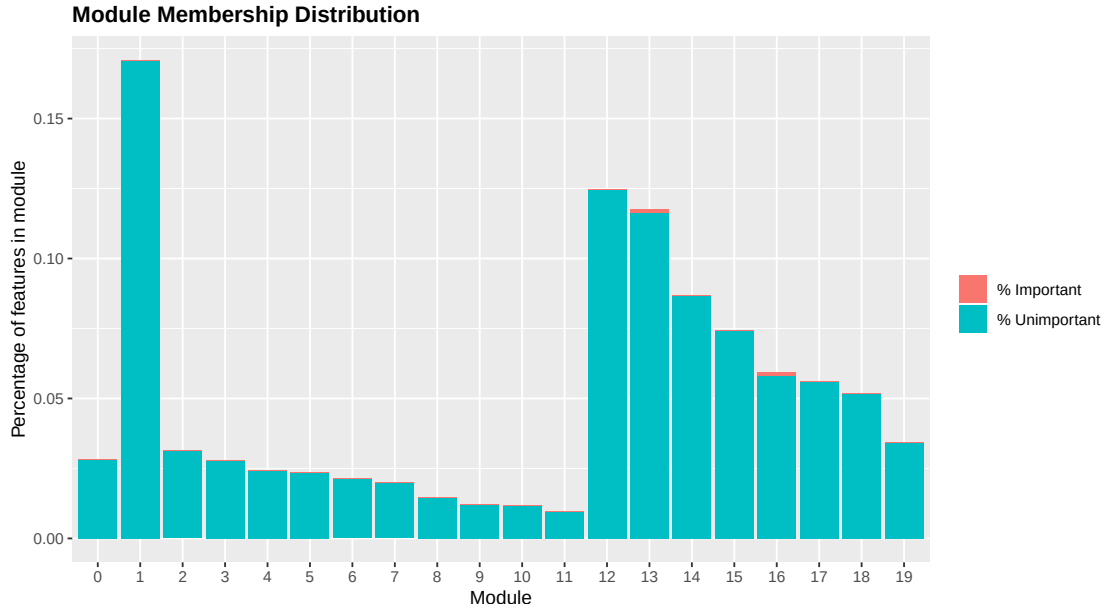

Figure 2: The height of the bars represents the proportion of features in each module. The proportion of each bar colored in red represents the proportion of features that are selected within each module.

The function `wff` returns an object of type `fuzzy_forest`. Objects of this type have the usual generic methods. The function `print` prints the selected features as well as module memberships. The function `predict(fuzzy_forest, new_data)` takes in a `data.frame` or `matrix` and produces predictions based on the selected features. The generic `predict` method for fuzzy forests produces a vector of predicted values for the set of observations in the data set used to fit fuzzy forests or, if a new, independent data set is provided as the value of the argument `new_data`, predicted values for observations in the new data set. Although the fuzzy forests algorithm was designed with feature selection in mind, it is possible to fit random forests using the selected features and to use the resulting model for prediction.

A `data.frame` with the selected features can be obtained by accessing `feature_list` from the `fuzzy_forest` object.

```
> wff_fit$feature_list
```

```
   feature_name variable_importance module_membership
4    MMT00026944            6.451356                 6
3    MMT00019254            6.265534                 6
7    MMT00067823            4.438564                 3
1    MMT00006001            4.148521                 3
9    MMT00074983            3.515628                 7
6    MMT00065159            3.150476                 3
8    MMT00070342            3.032299                 3
2    MMT00015534            2.959381                 6
5    MMT00061313            2.770847                 3
10   MMT00078732            2.605921                 6
```

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 on a validation 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 important. The height of the bars represents what percentage of the total p features fall into a particular module. The area of the bar which is red represents the percentage of each module that is selected by fuzzy forests. Applying the function `modplot` to the object `wff_fit` above, we obtain the graph in Figure 2.

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 and conditional inference forests when the features are correlated. We carry out two simulations. In the first simulation, data is generated from a linear model. In the second simulation, data is generated from a nonlinear model. For random forests, feature selection is carried out by selecting the features with top 10 permutation VIMs. For conditional inference forests, feature selection is carried out by selecting the features with top 10 conditional VIMs. The first simulation is closely related to the simulations given in Nicodemus *et al.* (2010) and Strobl *et al.* (2008), the key distinction being that the simulation below includes additional "noise" features that are unrelated to the outcome.

In all simulations, X_i is generated from a multivariate normal distribution. The error terms, ϵ_i, are normal with mean 0 and standard deviation 0.5. The marginal distribution of each feature is standard normal. Features are subdivided into distinct modules. The correlation between features in different modules is 0. If the features in a module are correlated with one another, the correlation between features within the same module is set to 0.8. In each simulation, features in the final module are independent of each other (i.e., with correlation 0).

For the simulation from a linear model, we carry out two simulation scenarios. For the first scenario, the number of parameters p is set to 100. In this scenario, there are 4 modules. The features in the 4th module are independent of each other and of the features in the other modules. The features in the other three modules are correlated with one another. Namely, $\{X^{(1)}, \ldots, X^{(25)}\}$, $\{X^{(26)}, \ldots, X^{(50)}\}$, and $\{X^{(51)}, \ldots, X^{(75)}\}$ constitute 3 distinct modules each containing 25 features. The final module is $\{X^{(76)}, \ldots, X^{(100)}\}$. In this scenario, $Y_i = X_i^\top \gamma + \epsilon_i$ and among the correlated features 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 are set to 0. In addition, the intercept term, γ_0, is set to 0.

To evaluate the feature selection performance, we compute the proportion of times the non-zero features were selected over 100 simulation runs. The results are displayed in Figure 3.

For random forests and conditional inference forests the results of this simulation are largely in line with the results from Nicodemus *et al.* (2010) and Strobl *et al.* (2008). Random forests is much less likely to select independent covariates than conditional inference forests. Fuzzy forests select important features with slightly lower frequency than conditional inference

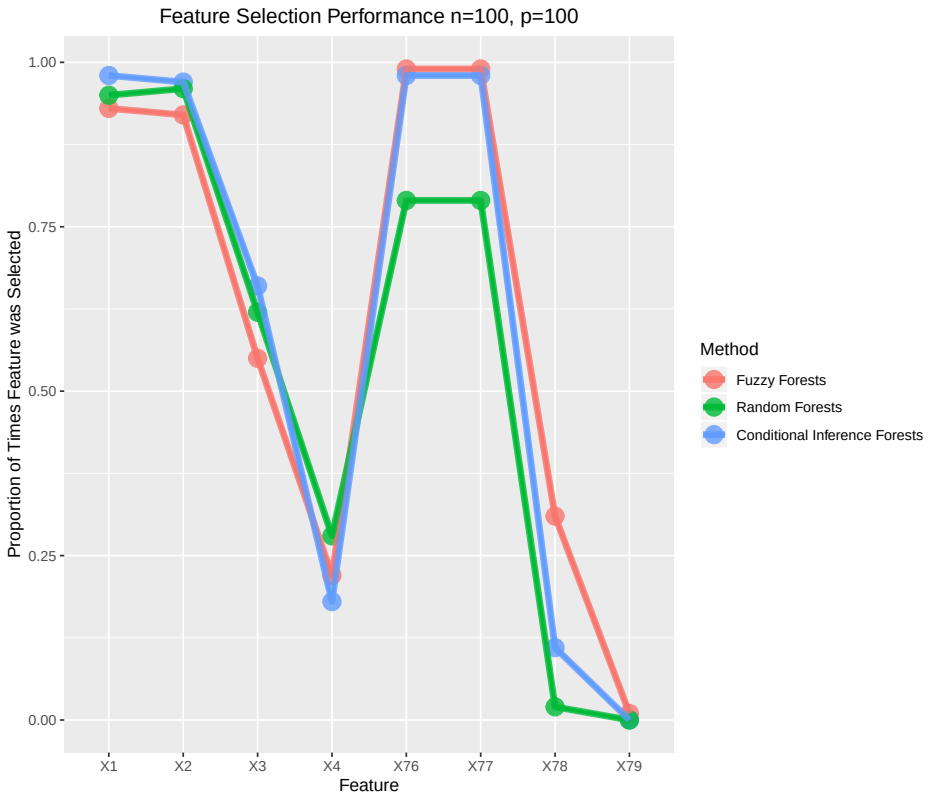

Figure 3: Fuzzy forests are compared to random forests with $p = 100$ and $n = 100$. The height of each point represents the proportion of times each feature was selected in 100 simulations. $X_1, X_2, X_3, X_{76}, X_{77}$, and X_{78} are important features. All other features were not important. X_1, X_2, and X_3 are correlated features. X_4 is correlated with X_1, X_2, and X_3 but is not important. X_{79} is independent but is not important. X_1, X_2 X_{76}, and X_{77} all have the same importance. X_3 and X_{78}, equally important, have lower importance than the other important features.

forests, however its performance is generally comparable.

In the second scenario for the linear model, we have the same setup as before except we increase p to 1,000 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, $\{X^{(901)}, \ldots, X^{(1,000)}\}$, 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 4, when $p = 1,000$, random forests permutation VIMs largely ignore the independent features.

For the second simulation in which data was generated from a nonlinear model, we set $p = 100$ and let n vary from 250 to 500. The correlation structure in this simulation is identical to correlation structure described above for the linear simulation with $p = 100$. The true regression model,

$$f(X) = X_1 + X_2 + 2.92X_1X_2 + \sqrt{15}X_3 + X_4^3 + X_{76} + X_{77} + 3.74X_{76}X_{77} + \sqrt{15}X_{78} + X_{79}^3,$$

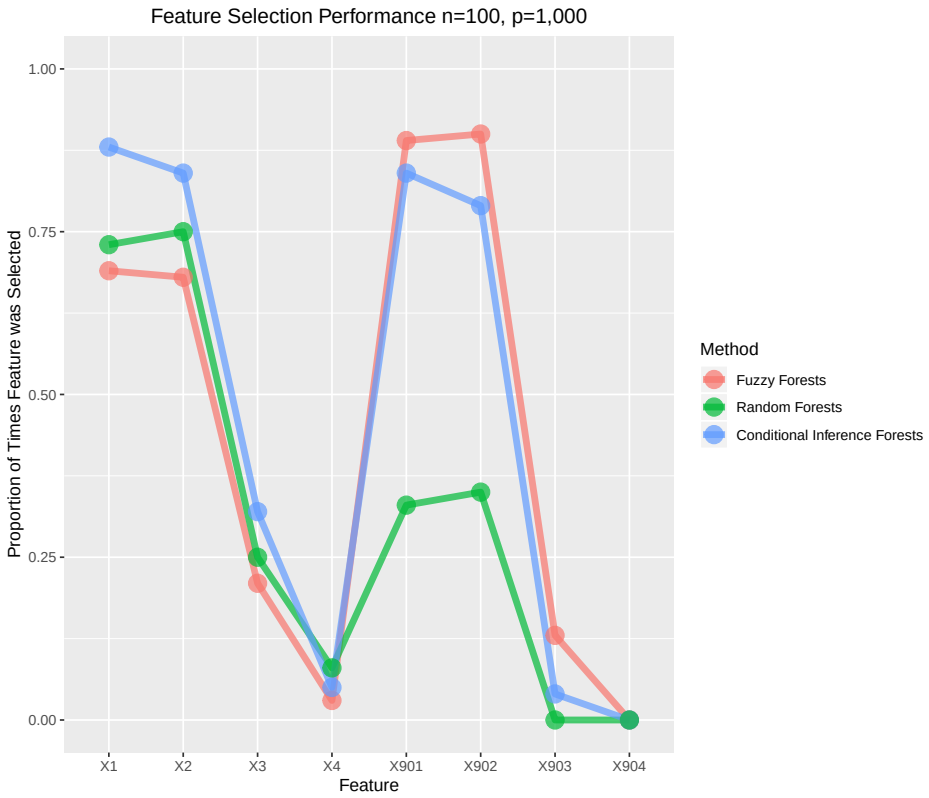

Figure 4: Fuzzy forests are compared to random forests with $p = 1,000$ and $n = 100$. The height of each point represents the proportion of times each feature was selected in 100 simulations. $X_1, X_2, X_3, X_{901}, X_{902}$, and X_{903} are important features. All other features were not important. X_1, X_2, and X_3 are correlated features. X_4 is correlated with X_1, X_2, and X_3 but is not important. X_{904} is independent but is not important. X_1, X_2, X_{901}, and X_{902} all have the same importance. X_3 and X_{903}, equally important, have lower importance than the other important features.

was designed such that the true VIM for each of the features upon which f depends are approximately equal to 30 (the true VIMs were calculated via Monte Carlo simulations). Therefore all of the features should be selected with equal probability.

For the nonlinear scenario with $n = 250$, we were able to compute VIMs for random forests and fuzzy forests, as well as conditional VIMs. For the scenario with $n = 500$, we were unable to compute conditional VIMs as the computational burden was too great. In our experience, the computational burden of calculating conditional VIMs increases more quickly with the sample size as opposed to the number of covariates.

The results of the first nonlinear scenario are displayed in Figure 5. First of all, note that none of the methods select features with equal probability, even within modules. In general, the features that are part of interaction terms (X_1, X_2, X_{76}, and X_{77}), are chosen with lower probability than the other 4 important features. All of these tree-based method have more difficulty detecting interactions in comparison to the linear and cubic terms, even conditional inference forests.

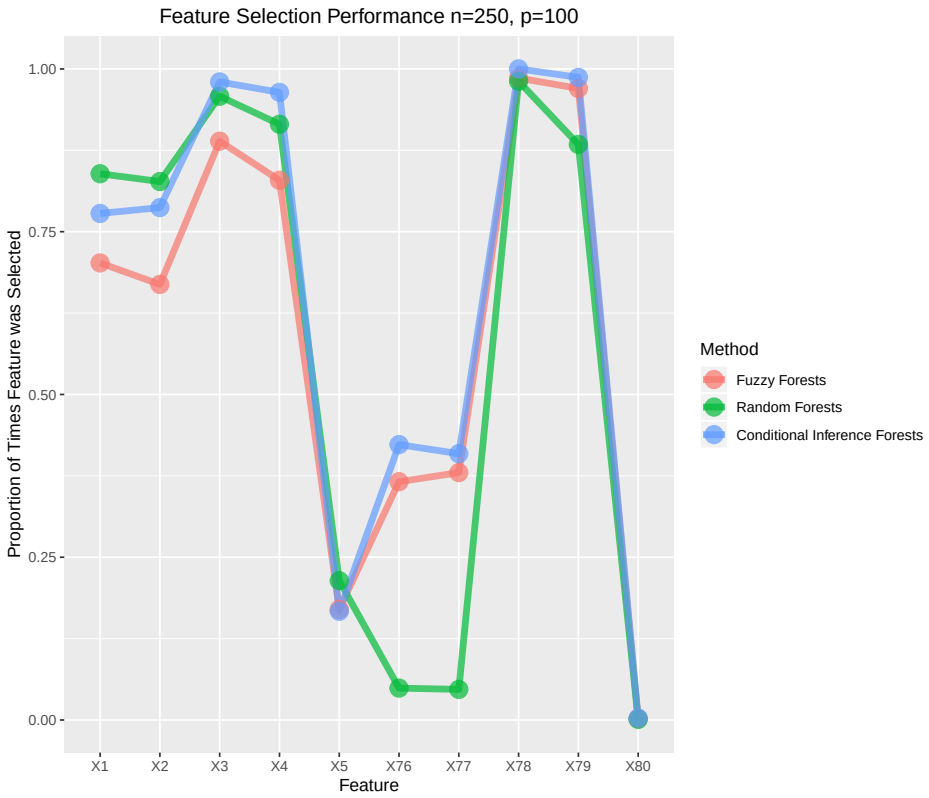

Figure 5: Fuzzy forests are compared to random forests with $p = 100$ and $n = 250$. The height of each point represents the proportion of times each feature was selected in 100 simulations. $X_1, X_2, X_3, X_4, X_{76}, X_{77}$, X_{78}, and X_{79} are important features. All other features were not important. X_1, X_2, X_3, X_4 are correlated with one another. X_5 is correlated with X_1, X_2, X_3, X_4 but is not important. X_{80} is independent and not important.

As in the first nonlinear simulation scenario, the correlated features are favored over the independent features. In particular, although both X_5 and X_{80} have VIM of 0, X_5 is selected with higher probability. Random forests are most heavily biased in favor of the correlated features and were largely unable to detect the interacting features X_{76} and X_{77}. Fuzzy forests perform slightly worse than both random forests and conditional inference forest on the correlated features, however, they perform comparably to conditional inference forests on the independent features. Overall, conditional inference forests seem to yield the best performance.

The results of the second scenario with $n = 500$ are displayed in Figure 6. As previously mentioned, calculation of conditional VIMs are computationally too burdensome. In this scenario, both random forests and fuzzy forests are able to select the correlated interacting features with higher probability (fuzzy forests, with smaller probability). Fuzzy forests also improve in its ability to select the independent interacting features with $n = 500$, while random forests are still largely unable to select these features.

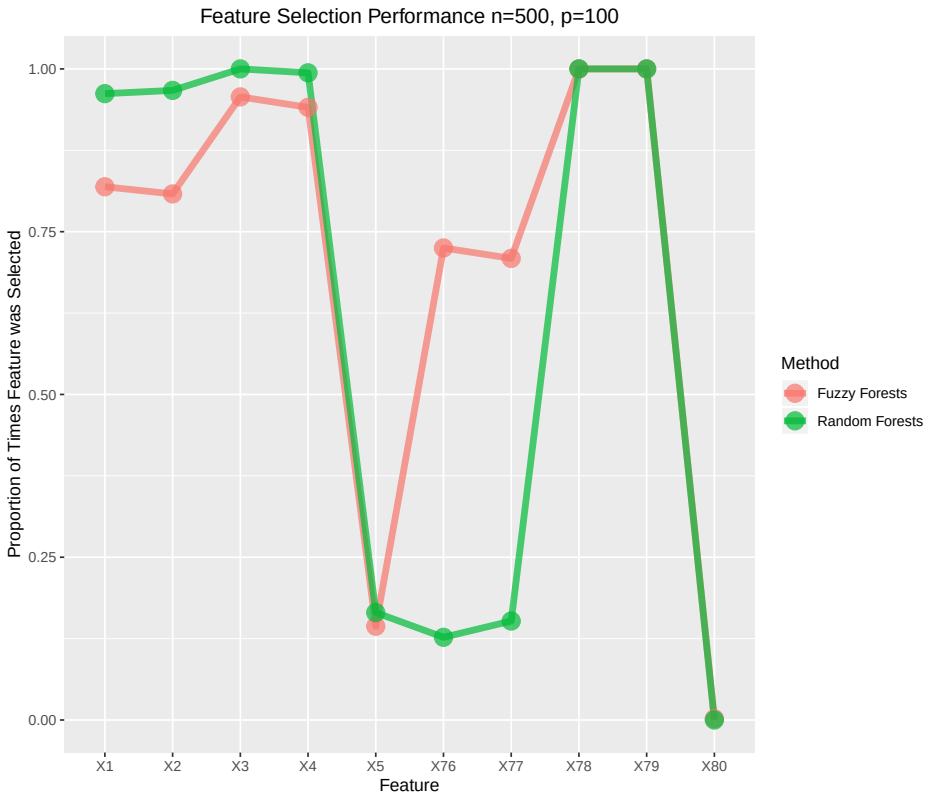

Figure 6: Fuzzy forests are compared to random forests with $p = 100$ and $n = 500$. The height of each point represents the proportion of times each feature was selected in 100 simulations. $X_1, X_2, X_3, X_4, X_{76}, X_{77}$, X_{78}, and X_{79} are important features. All other features were not important. X_1, X_2, X_3, X_4 are correlated features. X_5 is correlated with X_1, X_2, X_3, X_4 but is not important. X_{80} is independent but is not important.

5. Application

We demonstrate an application of **fuzzyforest** by using it to discover immunologic profiles that predict if an HIV patient will be able to control the virus without antiretroviral therapy (ART). An immunologic controller is defined as a patient able to achieve undetectable levels of the virus (< 50 copies/ml) without ART. Similarly, an immunologic responder is an aviremic patient, on ART, with sustained undetectable levels of the virus and CD4+ T cell counts above 350 cells/mm^3.

In this dataset there were 125 immunologic responders, 92 controllers ($n = 217$), and 313 features ($p = 313$). The features are derived from flow cytometry measurements. Flow cytometry may be used to measure the presence of various markers on the surface of a cell. The presence of up to 14 cell surface markers was measured. This yields up to 2^{14} possible binary combinations of markers, however, not all of these combinations were available. These markers assess immunological factors such as T cell maturation, activation, dysfunction, senescence, antigen-specificity and proliferation.

Features derived from flow cytometry measurements typically describe what proportion of cells in a sample display a subset of the aforementioned 14 markers. The presence of a

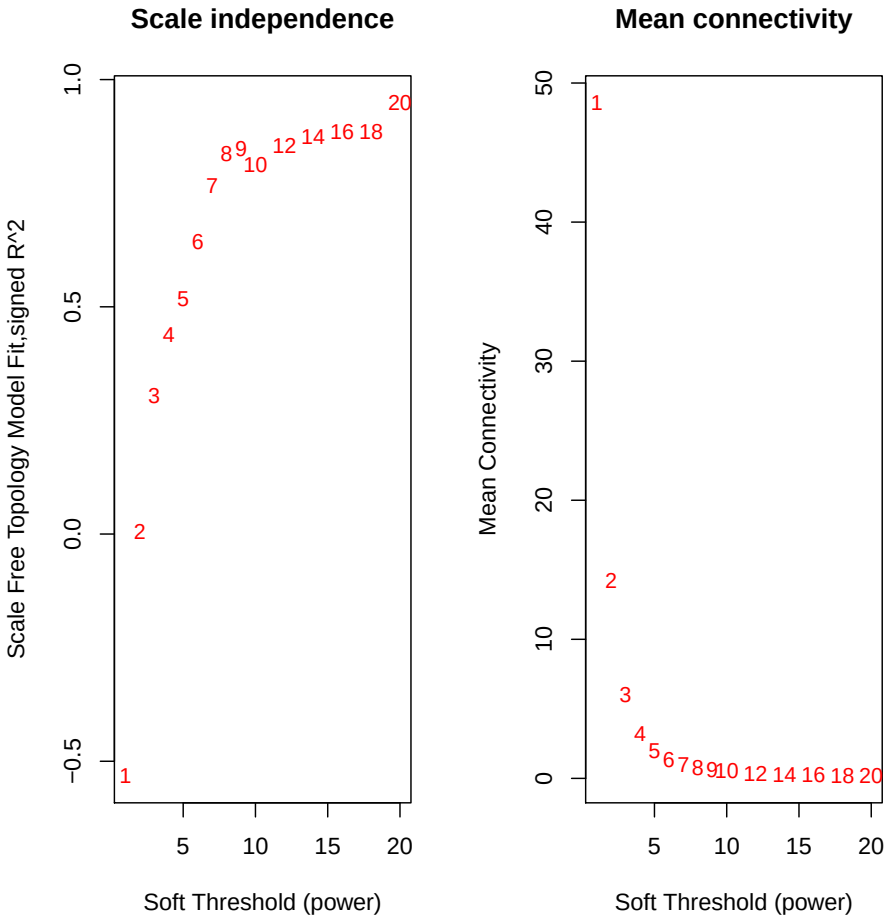

Figure 7: The following plot shows that 8 is the smallest power such that the scale free topology criterion is approximately met.

cell surface marker is denoted by + and the absence of the marker is denoted by −. For example, one feature may measure the proportion of all lymphocytes in a sample that are CD4 positive (CD4+). A second feature may measure the proportion of lymphocytes that display both CD4 and CD38 (CD4+CD38+). Because the group of CD4+CD38+ T cells is nested within the group of CD4+ T cells, the proportion of lymphocytes that are CD4+ will be positively correlated the proportion of lymphocytes that are CD4+CD38+. The nested nature of different subgroups of lymphocytes leads to high levels of correlation between features.

For some markers, mean florescence intensity, a continuous measure of the extent to which a cell displays a particular marker, was also measured using flow cytometry.

We used **WGCNA** to partition the features into modules. We used the scale-free topology criterion to determine the power of the adjacency function. We set $\beta = 8$ based on the elbow of the curves in Figure 7. We found 11 modules. Each modules is identified with a color. The choice of color is chosen randomly with the exception of the grey module. The grey module consists of features that are independent of the other modules. In our analysis, the largest module was the grey module with 140 features. It is commonly the case that the grey module is larger than the other modules. The smallest module, purple, was of size 10.

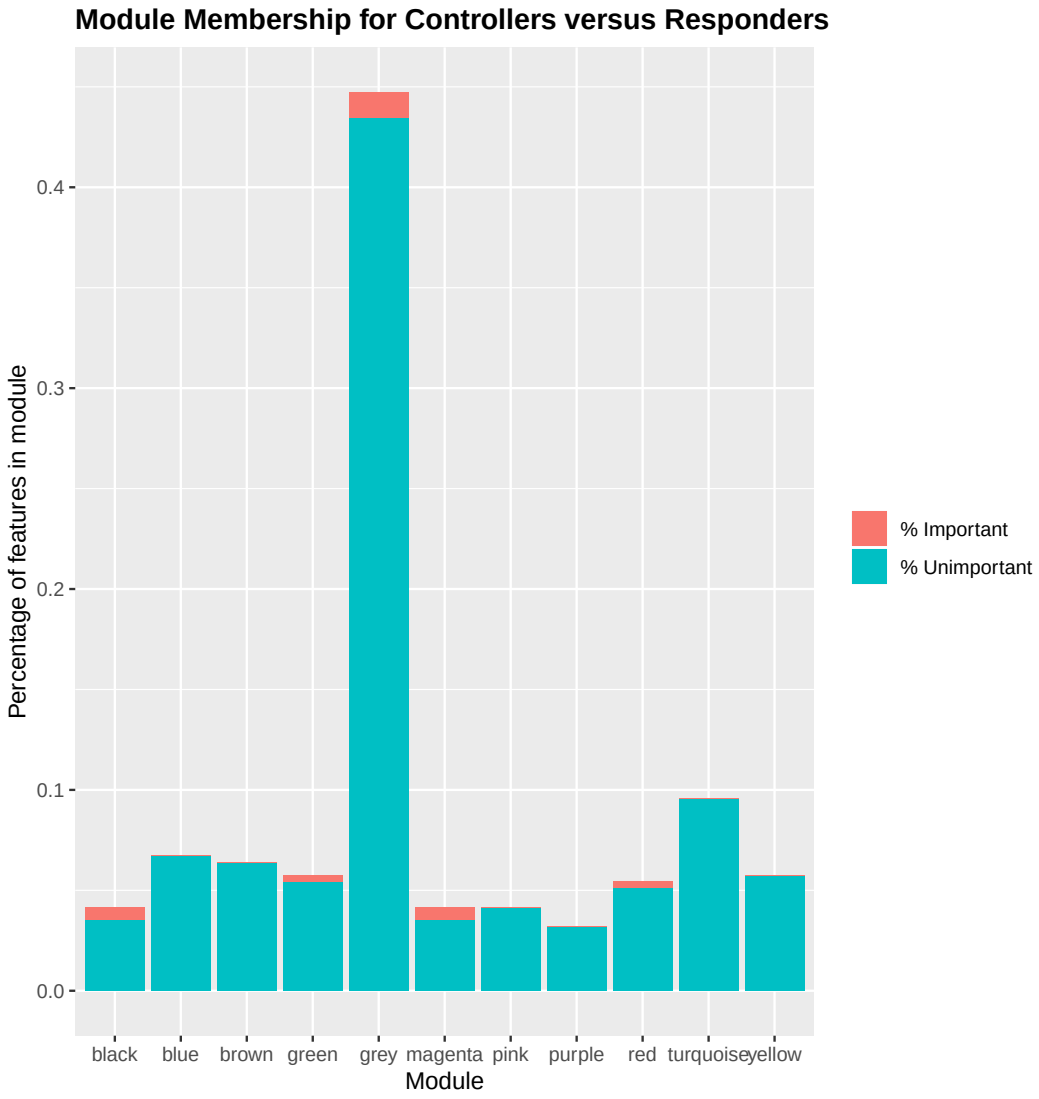

Figure 8: The height of the bars represents the proportion of features in each module. The proportion of each bar colored in red represents the proportion of features that are selected as important within each module.

We used the resulting module 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 for `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 immune activation; see Figure 9. The immune systems of the controllers are highly reactive to proteins specific to HIV, i.e., gag. The selection of cell surface markers such as PD-1 suggests that controllers may have a higher percentage of T cells that may be dysfunctional. It is notable that, while controllers had overall higher levels of immune activation (Hunt *et al.* 2008), they had lower activation in CD4+ central memory cells (Ramirez *et al.* 2016), as seen

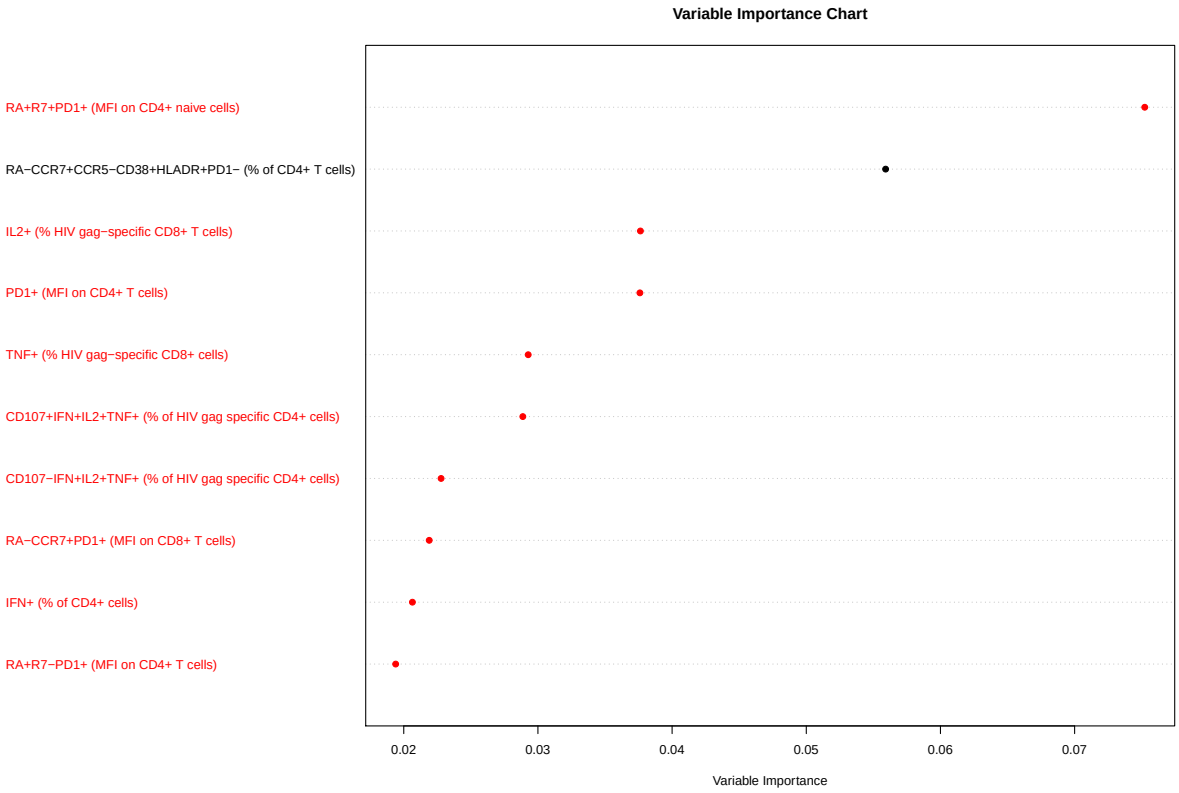

Figure 9: The following plot displays the importance of the top 10 selected features after fitting fuzzy forests. The variables are ranked from top to bottom. Red features indicate that controllers have higher values for the feature compared to responders. Black features indicate that responders have lower values for the feature compared to controllers.

in the feature ranked 2nd. These results are consistent with the nature of HIV pathogenesis. Indeed, it has been shown that limited infection of the central memory compartment is associated with lack of disease progression even in individuals who have detectable viremia (Klatt *et al.* 2014).

6. Discussion

In this article 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 in a computationally feasible manner, especially when $p \gg n$. Under these conditions, fuzzy forests are expected to outperform random forests. We found that, as expected, random forest VIMs were biased in favor of correlated features. Indeed when $p = 1,000$ while $n = 100$, random forests essentially ignored the independent variables that were important in the true model whereas fuzzy forests found them. The fuzzy forests algorithm is useful for screening large numbers of features or when it is desirable to find the most important features contributing to the signal.

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

We then used fuzzy forests to investigate immunologic phenotypes of patients who can control the virus without antiretrovirals. 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. It is expected that fuzzy forests will be useful in a wide variety of applications 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, AMFAR 8721SC, NIH T32 AI 007370, P30 AI 028697, and P01 GM 081621.

References

Breheny P, Huang J (2011). "Coordinate Descent Algorithms for Nonconvex Penalized Regression, with Applications to Biological Feature Selection." *The Annals of Applied Statistics*, **5**(1), 232–253. doi:10.1214/10-aoas388.

Breiman L (2001). "Random Forests." *Machine Learning*, **45**(1), 5–32. doi:10.1023/a:1010933404324.

Conn D, Ngun T, Ramirez CM (2019). ***fuzzyforest**: Fuzzy Forests*. R package version 1.0.6, URL https://CRAN.R-project.org/package=fuzzyforest.

Díaz-Uriarte R (2007). "**GeneSrF** and **varSelRF**: A Web-Based Tool and R Package for Gene Selection and Classification Using Random Forest." *BMC Bioinformatics*, **8**(1), 328. doi:10.1186/1471-2105-8-328.

Díaz-Uriarte R, Alvarez de Andrés S (2006). "Gene Selection and Classification of Microarray Data Using Random Forest." *BMC Bioinformatics*, **7**(1), 3. doi:10.1186/1471-2105-7-3.

Dua S, Acharya UR, Dua P (2014). *Machine Learning in Healthcare Informatics*. Springer-Verlag. doi:10.1007/978-3-642-40017-9.

Fan J, Li R (2001). "Variable Selection via Nonconcave Penalized Likelihood and Its Oracle Properties." *Journal of the American Statistical Association*, **96**(456), 1348–1360. doi:10.1198/016214501753382273.

Friedman J, Hastie T, Tibshirani R (2010). "Regularization Paths for Generalized Linear Models via Coordinate Descent." *Journal of Statistical Software*, **33**(1), 1–22. doi:10.18637/jss.v033.i01.

Gregorutti B, Michel B, Saint-Pierre P (2017). "Correlation and Variable Importance in Random Forests." *Statistics and Computing*, **27**(3), 659–678. doi:10.1007/s11222-016-9646-1.

Hastie T, Tibshirani R, Wainwright M (2015). *Statistical Learning with Sparsity: The Lasso and Generalizations*. Monographs on Statistics & Applied Probability. Chapman & Hall/CRC.

Horvath S (2011). *Weighted Network Analysis: Applications in Genomics and Systems Biology*. Springer-Verlag. doi:10.1007/978-1-4419-8819-5.

Hothorn T, Bühlmann P, Dudoit S, Molinaro A, Van der Laan MJ (2006). "Survival Ensembles." *Biostatistics*, **7**(3), 355–373. doi:10.1093/biostatistics/kxj011.

Hunt PW, Brenchley J, Sinclair E, McCune JM, Roland M, Shafer KP, Hsue P, Emu B, Krone M, Lampiris H, Douek D (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. doi:10.1086/524143.

Jiang H, Deng Y, Chen HS, Tao L, Sha Q, Chen J, Tsai CJ, Zhang S (2004). "Joint Analysis of Two Microarray Gene-Expression Data Sets to Select Lung Adenocarcinoma Marker Genes." *BMC Bioinformatics*, **5**(1), 81. doi:10.1186/1471-2105-5-81.

Klatt NR, Bosinger SE, Peck M, Richert-Spuhler LE, Heigele A, Gile JP, Patel N, Taaffe J, Julg B, Camerini D, Torti C, Martin JN, Deeks SG, Sinclair E, Hecht FM, Lederman MM, Paiardini M, Kirchhoff F, Brenchley JM, Hunt PW, Silvestri G (2014). "Limited HIV Infection of Central Memory and Stem Cell Memory CD4+ T Cells Is Associated with Lack of Progression in Viremic Individuals." *PLoS Pathogens*, **10**(8), e1004345. doi:10.1371/journal.ppat.1004345.

Langfelder P, Horvath S (2008). "**WGCNA**: An R Package for Weighted Correlation Network Analysis." *BMC Bioinformatics*, **9**(1), 559. doi:10.1186/1471-2105-9-559.

Langfelder P, Horvath S (2012). "Fast R Functions for Robust Correlations and Hierarchical Clustering." *Journal of Statistical Software*, **46**(11), 1–17. doi:10.18637/jss.v046.i11.

Liaw A, Wiener M (2002). "Classification and Regression by **randomForest**." *R News*, **2**(3), 18–22.

Mumford JA, Horvath S, Oldham MC, Langfelder P, Geschwind DH, Poldrack RA (2010). "Detecting Network Modules in fMRI Time Series: A Weighted Network Analysis Approach." *NeuroImage*, **52**(4), 1465–1476. doi:10.1016/j.neuroimage.2010.05.047.

Nicodemus KK, Malley JD (2009). "Predictor Correlation Impacts Machine Learning Algorithms: Implications for Genomic Studies." *Bioinformatics*, **25**(15), 1884–1890. doi:10.1093/bioinformatics/btp331.

Nicodemus KK, Malley JD, Strobl C, Ziegler A (2010). "The Behaviour of Random Forest Permutation-Based Variable Importance Measures under Predictor Correlation." *BMC Bioinformatics*, **11**(1), 110. doi:10.1186/1471-2105-11-110.

Ramirez CM, Sinclair E, Epling L, Lee SA, Jain V, Hsue P, Hatano H, Conn D, Hecht FM, Martin JN, McCune JM, Deeks SG, Hunt PW (2016). "Immunologic Profiles Distinguish Aviremic HIV-Infected Adults." *AIDS*, **30**(10), 1553–1562. doi:10.1097/qad.0000000000001049.

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

R Core Team (2019). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.

Strobl C, Boulesteix AL, Kneib T, Augustin T, Zeileis A (2008). "Conditional Variable Importance for Random Forests." *BMC Bioinformatics*, **9**(1), 307. doi:10.1186/1471-2105-9-307.

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. doi:10.1186/1471-2105-8-25.

Tibshirani R (1996). "Regression Shrinkage and Selection via the Lasso." *Journal of the Royal Statistical Society B*, pp. 267–288. doi:10.1111/j.2517-6161.1996.tb02080.x.

Van der Laan MJ (2006). "Statistical Inference for Variable Importance." *The International Journal of Biostatistics*, **2**(1). doi:10.2202/1557-4679.1008.

Zhang B, Horvath S (2005). "A General Framework for Weighted Gene Co-Expression Network Analysis." *Statistical Applications in Genetics and Molecular Biology*, **4**(1). doi:10.2202/1544-6115.1128.

Zhu R, Zeng D, Kosorok MR (2015). "Reinforcement Learning Trees." *Journal of the American Statistical Association*, **110**(512), 1770–1784. doi:10.1080/01621459.2015.1036994.

Affiliation:

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

Journal of Statistical Software http://www.jstatsoft.org/
published by the Foundation for Open Access Statistics http://www.foastat.org/
October 2019, Volume 91, Issue 9 *Submitted:* 2016-03-22
doi:10.18637/jss.v091.i09 *Accepted:* 2018-04-09