Journal of Statistical Software

MMMMMM YYYY, Volume VV, Issue II. *http://www.jstatsoft.org/*

Fuzzy Forests: A New WGCNA Based Random Forest Algorithm 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 Ramirez
UCLA SPH, Biostatistics

Abstract

In this paper we introduce fuzzy forests, a new machine learning algorithm for ranking the importance of features in high-dimensional classification and regression problems. Fuzzy forests is specifically designed to provide unbiased rankings of variable importance in the presence of correlated features. Fuzzy forests uses Weighted Gene Coexpressionn Network Analysis (WGCNA) to detect groups of highly correlated features. Unbiased rankings are obtained by fitting separate random forests on each module. We also introduce our implementation of fuzzy forests in the R package, **fuzzyforests**.

Keywords: Random Forests, WGCNA, machine learning,R.

1. Introduction

The problem of identifying important features in the presence of correlation has been an area of intense research within the statistics and machine learning community. Biological applications have, in particular, spurred the development of high dimensional feature selection methods. While model based feature selection algorithms such as the LASSO or SCAD may efficiently detect important features in presence of correlation (Raskutti *et al.* 2010), this efficiency

comes at the cost of making parametric assumptions that may not hold in practice.

Random forest variable importance measures (VIMs) offer a nonparametric alternative to model based feature selection algorithms (Breiman 2001). Random forests is a popular ensemble based machine learning algorithm. 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 forests VIMs are biased when features are correlated with one another ((Strobl *et al.* 2007); (Strobl *et al.* 2008); (Nicodemus and Malley 2009)).

Fuzzy forests cope with correlated features by taking a piecewise approach. We partition the set of features into distinct modules such that the correlation within each module is high and the correlation between groups is low. We then use an iterative feature selection algorithm to select the most important features from each. A random forest is then fit to the features that have survived this first round. A final iterative random forest, combining features from all modules, selects the top k features.

There are a variety of algorithms for partitioning features into distinct, high correlation modules. In this regard, WGCNA is our method of choice. WGCNA is a rigorous framework for detecting correlation networks (Zhang and Horvath 2005). Although it was first developed to detect modules of highly correlated genes, it has found application in a variety of biological contexts. The R package **WGCNA** is a robust, computationally efficient, and well-documented implementation of the WGCNA framework. 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 **WGCNA**'s fine documentation and tutorials.

The article is organized as follows. In section 2 of this article, we briefly review the random forests, WGCNA, and introduce the fuzzy forests algorithm. In section 3, we introduce the R package **fuzzyforest**. In section 4, we conduct simulations to comparing fuzzy forests to both random forests and conditional inference forests (CIFs). We demonstrate that fuzzy forests has performance comparable to that of CIFs' although at much lower computational cost. In section 5, we use fuzzy forests to determine which biological factors are important in determining how well an HIV patient copes with the virus. 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 define variable importance measures. We assume that our data comes in the form of n independently and identically distributed iid. pairs $(X, Y) \sim G$. Here X is p dimensional feature vector and Y is a scalar outcome. The value of the vth feature for the the ith subject will be denoted by $X_i^{(v)}$. The feature vector for the ith subject is denoted by $X_i = (X_i^{(1)}, \ldots, X_i^{(p)})$. Finally, let $X^{(v)} = (X_1^{(v)}, \ldots, X_n^{(v)})$ be set of values for feature v across all n subjects.

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

σ^2. In the regression setting, Y is unrestricted. In classification, Y is restricted to take the value 0 or 1.

If the goal is to predict a new outcome Y based off of measurements, X, a good estimate of $f(X)$ is all that is required. We are interested in more than prediction. We are interested in understanding how $f(X)$ changes as function of particular features. If the value of $f(X)$ varies widely according to a particular value of the vth feature, $X^{(v)}$ is, in some sense, an "important" in determinant of the outcome, Y.

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. If p is moderate or large, $f(X)$ is difficult to interpret. It is most common in this case, to assume $f(X)$ has a specific parametric form so that $f_\beta(X)$ is known up to a finite dimensional parameter β. In the case of linear regression, β is a vector of regression coefficients and we can measure the importance of one feature versus another feature by examining the absolute magnitude of their corresponding coefficents.

If the parametric model $f_\beta(X)$ is a close approximation to $f(X)$, it is possible that interpretations based off of $\hat{f}_{\hat{\beta}}(X)$ will not be misleading. Likewise, if $f_\beta(X)$ is a poor approximation of $f(X)$, the resulting interpretation will be misleading. The parametric approximation $f_\beta(X)$ may be inadequate for a variety of reasons. This may occur if important features are not observed. Even if all appropriate features are measured, $f_\beta(X)$ may fail to capture important interactions between features. If $f_\beta(X)$ is a linear regression model, $\sum_{v=1}^{p} \beta_v X^{(v)}$, the true $f(X)$ may be nonlinear in such a way that this best linear approximation fails to capture.

Permutation VIMs provide a means of summarizing the importance of individual features without making parametric assumptions. We define the permutation VIM of feature v, $X^{(v)}$, as

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

Here, $X_i^{(v)}$ and $\tilde{X}_i^{(v)}$ are iid with distribution $G_{X^{(v)}}$ where $G_{X^{(v)}}$ is the marginal distribution of $X_i^{(v)}$. This form of the VIM is given in a slightly different form in (Gregorutti *et al.* 2013) and (Zhu *et al.* 2012). These authors also discuss conditions under which the estimate of the permutation VIM derived from random forests is consistent.

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 and 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."

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 is highly unstable and gives highly variable predictions. Averaging multiple trees over many bootstrap samples leads to more stable estimates of $f(X)$. 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. Of these randomly chosen features,

the feature with the highest marginal importance is used to split the node. The number of randomly selected features at each stage is commonly denoted as $mtry$. High values of $mtry$ tend to lead to just a few important features getting selected at the majority cut-points. 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 $\sqrt{p}$. In the case of classification $\lfloor p/3 \rfloor$ is common choice.

Random forest VIMs are obtained by testing how predictive accuracy suffers when the values of individual predictors are permuted. Let $OOB_k \subset \{1, \ldots, n\}$ be the out of bag samples in the kth bootstrap sample. Let π_k be a random permutation of the elements of OOB_k and let $\pi_k^{(v)}(X_i) = (X_i^{(1)}, \ldots, X_{\pi_k(i)}^{(v)}, \ldots, X_i^{(p)})$, where $i \in OOB_k$. In other words, $\pi_k^{(v)}$ permutes the values of the vth feature across all out of bag subjects. The variable importance of the ith feature from the kth tree is defined as

$$\widehat{VIM}^k(v) = \frac{\sum_{i \in OOB_k} (y_i - \hat{f}^k(\pi_k^{(v)}(X_i))^2 - (y_i - \hat{f}^k(X_i))^2}{|OOB_k|} \tag{2}$$

The variable importance for the entire random forest is defined as

$$\widehat{VIM}(v) = \frac{\sum_{k=1}^{ntree} \widehat{VIM}^k(v)}{ntree} \tag{3}$$

2.3. A Brief Review of WGCNA

WGCNA is a rigorous framework for constructing a network of features. This network is then used to determine clusters or modules of inter-related features. We briefly review the steps of a WGCNA network analysis. The user first specifies a similarity function $s_{ij} = S(X^{(i)}, X^{(j)})$ 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 $|Corr(X^{(i)}, X^{(j)})|$ or $\frac{1+Corr(X^{(i)}, X^{(j)})}{2}$ according to whether the network is unsigned or signed.

This similarity matrix is then transformed into an adjacency matrix $A = [a_{ij}]$. This adjacency function determines how the adjacency function translates into network adjacencies. The simplest choice of adjacency function is the $signum(s_{ij}, \tau)$ function. This function simply sets a hard threshold τ. If $s_{ij} \geq \tau, a_{ij} = signum(s_{ij}, \tau) = 1$ otherwise $a_{ij} = 0$. Nodes are either connected or un-connected if the $signum$ adjacency function is used. In practice, a soft-thresholded network is often more plausible than a hard-thresholded one. The power function $a_{ij} = s_{ij}^q$ is common choice of soft-thresholding adjacency function. Large values of q yield behavior closer to a hard-thresholded network. Once an adjacency function is calculated, an hierarchical clustering tree algorithm may be used to define 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

$$\omega_{ij} = \frac{l_{ij} + a_{ij}}{min\{k_i, k_j\} + 1 - a_{ij}} \tag{4}$$

where $l_{ij} = \sum_u a_{iu} a_{uj}$ and $k_i = \sum_u a_{iu}$. The topological overlap between two nodes can be high even if a_{ij} 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.

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 $p(k) \sim k^{\gamma}$ or $\log_{10}(p(k)) \propto \log_{10}(k)$. This suggests that one should select a smaller value of q such that the R^2 between $\log_{10}(p(k))$ and $\log_{10}(k)$ 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 following section, we describe the motivation behind fuzzy forests and explain why it should provide less biased estimates of VIMs. In this section, we describe the algorithm.

The fuzzy forests algorithm can be subdivided into two steps: a screening step and a selection step. The screening step works in a piecewise fashion to screen out unimportant features. The screening step takes as input a partition of the features such that the correlation within each partition is roughly constant. Our package, **fuzzyforest**, facilitates the use of WGCNA to obtain such a partition of features. However, it is possible to use alternative methods to partition the features. Denote this partitioning of the features by 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, the screening step fits an iterative series of random forests to eliminate features in backwards stepwise fashion. Starting with all features in the partition P_l, a random forest is fit using all features in P_l. The least important features are then eliminated, call the reduced set of features after the first random forest $P_l^{(1)}$. For example, the features with VIM in bottom 25% might be dropped at each step. A 2nd random forest is fit using features in $P_l^{(1)}$. The least important features from this latest random forest are then eliminated leading a further reduced set of features $P_l^{(2)} \subset P_l^{(1)} \subset P_l$. Call the subset obtained after the kth iteration $P_l^{(k)}$ and let $p_l^{(k)} = |P_l^{(k)}|$. 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 P_l features remain.

The user must specify a few tuning parameters at the screening step. First of all, the user must specify how many features are to be dropped after each random forest is fit, 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 will be called the $keep\_fraction$. The first time the number of features drops below $keep\_fraction * p_l$, the iterative series of random forests will stop and the top $\lfloor keep\_fraction * p_l \rfloor$ features will be selected. More precisely, for the first iteration k such that $p_l^{(k)} < keep\_fraction * p_l$, we retain the top $\lfloor keep\_fraction * p_l \rfloor$ features from $P_l^{(k-1)}$.

For each random forest, $mtry$ and $ntree$ must appropriately selected. Since the number features varies across random forests, $mtree$ and $ntree$ must be a function of the current number of features. Suppose we at iteration k and are about fit a random forest to obtain $P_l^{(k+1)} \subset P_l^{(k)}$. In the case of regression, **fuzzyforest** lets $mtry = \sqrt{p_l^{(k)}} mtry\_factor$. For

classification **fuzzyforest** sets $mtry = \left\lfloor p_l^{(k)}/3 \right\rfloor mtry\_factor$. In both cases, $mtry\_factor$ must be pre-specified by the user with a default of 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 will take a long time to run. The package **fuzzyforest** sets $ntree = max(min\_ntree, p_l^{(k)} * ntree\_factor)$.

The selection step consists of one last iterative series of random forests. This series of random forests is fit on 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. The user specifies how many features will selected by the selection step.

2.5. Motivation for Fuzzy Forests Algorithm

The selection step of fuzzy forests is motivated by the following observation concerning the theoretical properties of VIMs. Let $G_{P^{(l)}}$ denote the joint distribution of the features in partition $P^{(l)}$ and let $X^{P^{(l)}} \sim G_{P^{(l)}}$. Suppose that $X^{P^{(i)}} \perp X^{P^{(j)}}$ for all $i \neq j$ and suppose that $f(X) = \sum_{j=1}^m f_j(X^{P^{(j)}})$. 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 $E[E[Y|X]|X^{(l)}] = E[f(X)|X^{P^{(l)}}] = \sum_{j=1}^m E[f_j(X^{P^{(j)}})|X^{P^{(l)}}] = f_l(X^{P^{(l)}}) + \sum_{j\neq l}^m E[f_j(X^{P^{(j)}})]$

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:

$$VIM^*(v) = E(f_l(X_i^{(l_1)}, \ldots, X_i^{(v)}, \ldots, X_i^{(l_m)}) - f_l(X_i^{(l_1)}, \ldots, \tilde{X}_i^{(v)}, \ldots, X_i^{(l_m)}))^2. \quad (5)$$

Here, $X_i^{l_k}$ is the kth element of partition $P^{(l)}$. As in equation (1), $X^{(v)}$ 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 are no longer competing against one another.

This above observations suggest that we should fit a separate random forest to each module and then report the the resulting estimates of the VIMs. Why does the fuzzy forests algorithm diverge from this simple procedure? We stress that the VIMs obtained by analyzing each module separately are, in general, different than the VIMs obtained by fitting a single random forests. The assumptions of additivity across modules and independence of modules are quite strong.

The fuzzy forests algorithm relaxes these overly restrictive assumptions. Fuzzy forests allows for interactions between features that were found to be important within modules. In biolog-

ical applications, modules might represent different biological systems. thus, it is reasonable to allow these systems to interact with one another. In fuzzy forests, the selection step acts as a screening mechanism. Fuzzy forests is implicitly making the assumption that the features that have high VIMs.

It is important that iterative random forests are used at the 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.

We do not recommend interpreting the estimated VIMs too closely. 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 quite biased.

3. The fuzzyforest package

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

The second function `fuzzyforest` assumes that the features have already been partitioned into separate modules. In particular a user is able to carefully conduct a separate module detection analysis using **WGCNA**. Such an analysis might find the smallest value of p such that the scale-free topology criterion is met. The function `fuzzyforest` is then 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` with constructor `WGCNA_control()`. Similarly tuning parameters related to the screening step and the select step are specified through objects of type `screen_control` and `select_control`.

We demonstrate the workings of **fuzzyforest** with a small simulation. The features have multivariate normal distribution and were generated using the package **mvtnorm**. There are 5 modules. Within each module the correlation between features is .8 and the variance of each feature is 1. The modules are independent of one another. The true model is linear and two of the 5 modules have 2 significant coefficients set to 5.

```
R> n <- 100
R> p <- 50
R> mod_sigma <- matrix(.8,10,10); diag(mod_sigma)=1
R> sigma <- as.matrix(.bdiag(lapply(1:5, function(j) mod_sigma)))
R> X <- rmvnorm(n, mean=rep(0, 50), sigma=sigma)
R> beta <- c(rep(5,2),rep(0,10-2),rep(5,2),rep(0,50-12))
R> y <- X%*%beta + rnorm(n, sd=.1)
R> X <- as.data.frame(X)
R> fit <- WGCNA_fuzzyforest(X,y,WGCNA_params=WGCNA_control(minModuleSize=1)
+ ,screen_params = screen_control(keep_fraction=.5, drop_fraction=.2)
```

```
+ ,select_params = select_control(number_selected=4, drop_fraction=.1)
+ ,num_processors=4)
```

If the outcome, `y`, is a `factor` the random forests will be constructed using classification trees, otherwise regression trees will be used. `WGCNA_fuzzyforest` returns an `S3` object of type `fuzzy_forest` consisting of a list. This list contains a `data.frame` with the ranking and VIM of the selected features. It also contains the final random forest that was fit at the selection step, along with the module membership and further information about the weighted co-expression network.

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 via a random forest based on the selected features.

The number of features selected by fuzzy forests must be pre-specified. It is possible to fit separate models selecting a different numbers of features for each model. The predictive accuracy on a test set can then be used to select one of these models. If a test set is unavailable a plot of the test error versus the number of features selected may shed light on when the fuzzy forests algorithm begins over-fitting. Note that these models will not necessarily be nested.

The generic function `plot` yields a visual display of which modules are 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 `plot` to the object `fit` above, we obtain the following graph.

4. Simulations

In this section we demonstrate the performance of fuzzy forests on a number of simulated data sets. Our first set of simulations assumes a logistic regression model. In all simulations, the sample size was $n = 1000$. We let the number of features equal $p = 100$ and 1000. For each simulation, we carried out 100 simulations. In each simulation, the number of features selected was 10.

Modules were generated using the function `simulateModule` from **WGCNA**. In the case, where $p = 100$ there are 3 modules each of size 25. The features were multivariate normal. Each feature had variance 1. The correlation within each module was .8. The modules were independent of one another and there were an additional 25 features that were independent of each other and independent of each module.

The true regression function is of the form $\log(p_i/(1-p_i)) = X_i'\beta$ where p_i is probability that subject i has outcome 1. For the case when $p = 100$, the first 3 features are significant with values 1, .5, and .25. The first 3 independent features are also significant with the same values. We therefore have

$$\log(p_i/(1-p_i)) = x_1 + .5x_2 + .25x_3 + x_{76} + .5x_{77} + .25x_{78} \tag{6}$$

where x_1, x_2 and x_3 are highly correlated and x_{76}, x_{77}, and x_{78} are independent.

We present below the results of fuzzy forests on 100 simulated data sets. In addition to presenting the VIMs for the significant predictors, we also present the VIMs for x_4 and x_{79}.

Note that fuzzy forests was able to detect the 4 most important features: x_1, x_2, x_{76}, and x_{77}. The simulation also demonstrates fuzzy forests still has a tendency to select unimportant features that are correlated with important ones. Although this bias is much reduced in comparison to random forests.

Table 1: Results of Simulation for $p = 100$

Feature	Avg VIM	Probability of Selection
x_1	.058	.75
x_2	.023	.32
x_3	.014	.21
x_4	.007	.10
x_{76}	.074	1.0
x_{77}	.062	.83
x_{78}	.001	.14
x_{79}	.000	0.0

In the case where $p = 1000$, there were 9 modules each of size 100. Modules were independent of one another and the correlation structure within modules was once again .8. The last 100 features were independent of one another and of the 9 modules. The first 3 of these independent features were significant yielding the following regression function:

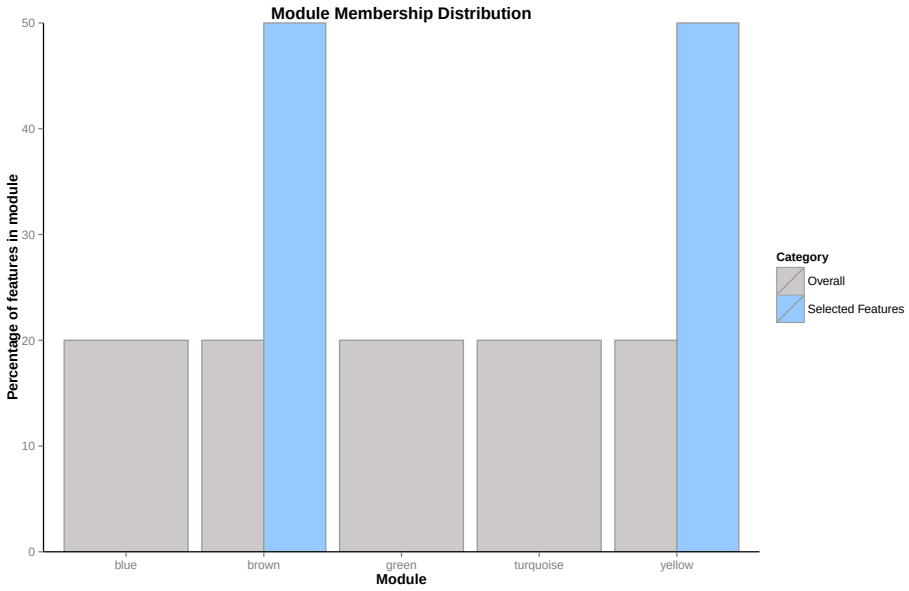
$$\log(p_i/(1-p_i)) = x_1 + .5x_2 + .25x_3 + x_{901} + .5x_{902} + .25x_{903} \tag{7}$$

We show the average VIMs over 100 simulations and present the VIMs of x_4 and the x_{904} in addition to the significant ones. Note that the VIMs are at an overall lower level. The selection probability of the most significant features remain somewhat similar to the case where $p = 100$. The selection probability of x_4 is lower than it was in the case of $p = 100$, however, the selection probabilities were overall lower in the simulation.

Table 2: Results of Simulation for $p = 1000$

Feature	Avg VIM	Probability of Selection
x_1	.023	.70
x_2	.007	.22
x_3	.004	.11
x_4	.000	.01
x_{901}	.032	1.0
x_{902}	.026	.82
x_{903}	.003	.08
x_{904}	.000	0.0

Figure 1: Result of `plot(fit)`



References

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

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

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

Journal of Statistical Software http://www.jstatsoft.org/
published by the American Statistical Association http://www.amstat.org/
Volume VV, Issue II *Submitted:* yyyy-mm-dd
MMMMMM YYYY *Accepted:* yyyy-mm-dd