Journal of Data Science 14(2016), 67-96

Understanding Variable Effects from Black Box Prediction: Quantifying Effects in Tree Ensembles Using Partial Dependence

Guy Cafri,Barbara A. Bailey

University of California, San Diego
San Diego State University

Abstract: Scientific interest often centers on characterizing the effect of one or more variables on an outcome. While data mining approaches such as random forests are flexible alternatives to conventional parametric models, they suffer from a lack of interpretability because variable effects are not quantified in a substantively meaningful way. In this paper we describe a method for quantifying variable effects using partial dependence, which produces an estimate that can be interpreted as the effect on the response for a one unit change in the predictor, while averaging over the effects of all other variables. Most importantly, the approach avoids problems related to model misspecification and challenges to implementation in high dimensional settings encountered with other approaches (e.g., multiple linear regression). We propose and evaluate through simulation a method for constructing a point estimate of this effect size. We also propose and evaluate interval estimates based on a non-parametric bootstrap. The method is illustrated on data used for the prediction of the age of abalone.

Key words: Bagging, Bootstrap, Data Mining, Random Forest, Partial Dependence

1. Introduction

Most science is concerned with characterizing the effect of one or more variables on an outcome, in particular the nature and strength of the relationship. Data mining methods are generally distinguished by a great deal of flexibility in the number of predictor variables than can be considered and how their effects are “modeled”, but it is often at the expense of being able to adequately characterize their effects. Traditional parametric models such as multiple linear regression require correct model specification in order for estimates to be trustworthy and become unreliable when the number of predictors (p) approaches the number of observations (n). In contrast, data mining approaches such as random forests (Breiman, 2001) do not require any model specification and can be used irrespective of the size of p relative to n. However, random forests suffer from an inability to characterize variable effects in a substantively meaningful way. By comparison, a regression coefficient from a multiple linear regression is informative to the extent that it conveys the strength and direction of a variable’s effect on the response (in the original scale of both variables), while controlling for the effect of other variables. The absence of methods to quantify variable effects in random forests in a similar manner is possibly one reason that they have not yet enjoyed widespread use in scientific applications, where interpretation of a numerical quantity that describes a variable's effect is of central importance (Dasgupta et al., 2012; Friedman, 2001). Even in research contexts where prediction might be the primary objective, it is often also desirable to be able to interpret variable effects using effect sizes with an interpretation similar to coefficients from a multiple linear regression. In this paper we address this problem by proposing an effect size based on the method of partial dependence (Friedman, 2001), which avoids problems related to model misspecification and implementation in high dimensional settings by using random forests as the basis of the estimation.

Tree-based methods are among the best for data mining applications in a predictive context, allowing for diverse inputs and an ability to identify relevant predictors from a large pool (Hastie, Tibshirani, and Friedman, 2009). Random forests (Breiman, 2001) represent an important advancement over prior tree-based methods in generating accurate predictions. The improved prediction accuracy in random forests is achieved by combining bagging (i.e., bootstrap aggregating (Breiman, 1996)) with random predictor selection at each tree split. Bagging reduces the variance through its averaging of identically distributed trees, an effect that is enhanced by random feature selection, which further reduces the variance by de-correlating the trees (Breiman, 2001; Hastie, Tibshirani, and Friedman, 2009). This improvement in prediction accuracy comes at the cost of interpretability however, because the simplicity of a single tree is replaced by an ensemble of trees.

To date several approaches have been described that could be used to interpret variable effects in random forests. One option is to use permutation importance (Hastie, Tibshirani, and Friedman, 2009) or conditional permutation importance (Strobl et al., 2008). Generally, permutation importance does not characterize variable effects in a way that is substantively meaningful, as it only describes the predictive ability of a variable relative to other variables in the forest. Another possibility is to identify the most important variables (e.g., via permutation importance) and to then use this predictor variable subset in a conventional parametric statistical model that can be used for variable interpretation. Although such an approach does convey variable effects in a substantively meaningful way given the availability of regression coefficients, it is somewhat unsatisfactory. Specifically, it may omit variables that are important and will not control for the effects of all variables when estimating the effects of variables that are in the subset, unless a rather inclusive variable retention strategy is adopted (Strobl, Malley, and Tutz, 2009). Furthermore, this approach relies on correct model specification of all variables. Yet another alternative is partial dependence (Friedman, 2001), which can be interpreted as the effect of one or more variables on the response (in their original scale), averaging over the effects of other variables used to grow the forest. Although this method is appealing given its similarity to interpretation of coefficients from a multiple linear regression, to date this method has been limited in application to graphical depictions. In this paper we describe a method that can be used to generate point estimates and confidence intervals based on the method of partial dependence. We propose and evaluate through simulation a method for constructing a point estimate and interval estimates based on a non-parametric bootstrap (Efron, 1987; Efron and Tibshirani, 1993) The method is illustrated using data for the prediction of the age of abalone.

2. Methodology

2.1. Random Forests

Random forests can be described in several steps. Adopting the notation of Hastie et al. (2009), the first step is to take B bootstrap samples of size N from the data, where N corresponds to the total number of observations in the data. Grow a tree T_b on each bootstrap sample for $b = 1, \ldots, B$. The tree growing process is based on what is typically used for decision trees (Breiman et al., 1984) except that at each split a subset, m, of the total number of variables, p, is selected at random. Furthermore, trees are not pruned, with terminal nodes occurring when the minimum node size is reached. Prediction for an observation x in the context of regression is based on taking the average value on the response for the terminal node in the bth tree where x appears, $\hat{T}_b(x)$, and averaging over all trees in the forest:

$$\hat{f}_{rf}^{B}(x) = \frac{1}{B}\sum_{b=1}^{B} \hat{T}_b(x) \quad (1)$$

In the context of classification, letting $\hat{C}_b(x)$ be the class prediction of the bth tree, the prediction is conventionally based on:

$$\hat{C}_{rf}^{B}(x) = majority\ vote\ \{\hat{C}_b(x)\}_1^B \quad (2)$$

In the case of two classes, if the relative frequency of the event class is greater than .5 in the terminal node of x for the bth tree, an event would be predicted, whereas if it is less than .5 then no event is predicted. Then, a tree-averaged prediction for an observation is based on whether or not the proportion of trees predicting an event exceeds .5.

2.2. Partial Dependence

Using the notation of Hastie et al. (2009), consider the subvector X_S of $l < p$ for the input predictor variables $X' = (X_1, X_2, \ldots X_p)$ with $S \subset \{1,2,\ldots,p\}$. Further, let C be the complement set with $S \cup C = \{1,2,\ldots,p\}$. X_S is a set of variables whose 'target' effect is of interest, and X_C is the set of all other variables in the data which we seek to average over. Partial dependence is then defined as (Friedman, 2001):

$$f_S(X_S) = \mathrm{E}_{X_C} f(X_S, X_C) = \int f(X_S, X_C) p_C(X_C)\, dX_C \quad (3)$$

with marginal probability density $p_C(X_C) = \int p(X)\ dX_S$, where $p(X)$ is the joint density over all inputs.

The quantity in (3) for a single tree in a forest is estimated by (Friedman, 2001):

$$\bar{f}_S(X_S) = \frac{1}{N}\sum_{i=1}^{N} f(X_S, x_{iC}) \tag{4}$$

where $x_{iC}, x_{2C}, \ldots, x_{NC}$ are the values of X_C with N observations. To obtain an estimate of partial dependence for a forest, we would average over the b trees in the forest:

$$\hat{Y}_d = \frac{1}{B}\sum_{b=1}^{B} \bar{f}_S(X_S)_b \tag{5}$$

where $d = 1, \ldots, D$, corresponding to the distinct combination of values taken on by X_S. Generally, X_S can be comprised of multiple variables, each variable having two or more unique values. In this general case D would correspond to the total number of distinct combinations of values across those variables. The utility of considering multiple variables in a partial dependence approach, which might be referred to as 'multivariable partial dependence', is that it allows the possibility of examining interactions among two or more variables while averaging over the effects of all other inputs. More common however is to consider only a single variable partial dependence (hereafter the only type of partial dependence that is considered). In this case D would correspond simply to the number of distinct values a single variable takes on in a dataset. To obtain a prediction for a particular value, $X_S = x$, from an individual tree that has been grown, all observations in the dataset are assigned the value of x while keeping the values of all other variables as they are. This synthetic data is then passed through the tree to construct the prediction. To obtain the forest averaged prediction for x, do the same for the remaining B trees and take an average of the predictions. Finally, repeat this process for all other values of X_S occurring in the data. Typically, partial dependence plots have distinct values of X_S plotted on the X-axis and their forest-averaged prediction on the Y-axis.

2.3. An Effect Size Based on Partial Dependence

The general idea of the proposed approach is to use the output produced as part of partial dependence to fit a parametric model, which in turn is used to obtain a point estimate. It is important to note that partial dependence for a single variable will only have the functional form for that variable specified (if that variable is nominal and treated as a categorical input then no restriction on functional form is placed), all the remaining variable inputs which are being averaged over are not subject to parametric restrictions, one of the distinct advantages of using random forests. Of course any misspecification of functional form can be mitigated by fitting a model that corresponds to the form observed in a partial dependence plot.

For continuous outcome data a point estimate is based on the regression coefficient from a weighted least squares regression, with weights corresponding to the frequency with which each distinct value of X_d, $d = 1 \ldots D$, appears in the original dataset. In this case the model is:

$$\hat{Y}_d = X_d\beta + \varepsilon_d \tag{6}$$

with $\hat{\beta} = (X'\Omega^{-1}X)^{-1}X'\Omega^{-1}\hat{Y}$

and $\Omega = diag\ (n_1, \dots, n_D)$ where n is a frequency

β from above can be interpreted as an average treatment effect (ATE) (Imbens, 2004). This is the effect, in the population, of moving all subjects from being untreated to treated (i.e., in the case of a binary explanatory variable characterized by the presence or absence of treatment). The ATE interpretation follows from the manner in which the outcome, $\hat{Y}_d$, is calculated. That is, predictions are averaged over all observations for each level of an explanatory variable.

Confidence intervals for $\hat{\beta}$ can be based on a nonparametric bootstrap. One option is the percentile method (Efron, and Tibshirani, 1993). A two-sided confidence interval is calculated by using the sorted bootstrap distribution of the estimated effect, $\hat{\beta}^*$, to identify the values of the lower and upper bound given B bootstrap replicates:

$$(\hat{\beta}^*_{B(\alpha/2)}, \hat{\beta}^*_{B(1-(\frac{\alpha}{2}))}) \tag{7}$$

A better interval is the bias-corrected and accelerated confidence interval (BC$_a$), which corrects for bias and skewness in the shape of the bootstrap distribution (Efron, 1987).

To summarize, the proposed algorithm consists of the following:

1) Grow a forest.
2) Estimate partial dependence (for a single variable).
   a. Create D datasets $(d = 1, \dots, D)$. For all observation in the d^{th} dataset only let them take on one value for the variable of interest while keeping values of all other variables unchanged.
   b. Pass the d^{th} dataset through each tree and average the predictions over the trees in the forest.
   c. Repeat Part b for each of the D datasets
3) Construct a point estimate of the proposed effect size by fitting a weighted least squares model with response based on the tree-averaged predicted values obtained in Step 2, the explanatory variable corresponding to the value used to generate each tree-averaged prediction, and weight based on the frequency each value the explanatory variable takes on in the original data.
4) For confidence intervals, repeat Steps 1-3 for as many bootstrap samples as desired

3. Numerical Examples

3.1. Simulation Design

The design of the simulation is based on manipulating the magnitude of the target variable's effect and sample size. We used 10,000 simulated cases per condition to evaluate bias and overall accuracy (root mean square error (RMSE)) of the point estimate, and 1,000 simulated cases per condition to evaluate interval estimates with 1,000 bootstrap replicates per simulated case. Bias was calculated as the parameter minus the estimate, averaged over the number of simulations. The RMSE was calculated by taking the square root of the average squared distance of the estimate from the parameter. Coverage was calculated as the proportion of times the parameter is captured by the 95% bootstrap confidence interval.

Data Generation Process 1

Data generation process (DGP) 1 is based on the following model:

$$Y_i = \beta_0 X_{i0} + \beta_1 X_{i1}^* + \cdots + \beta_{12} X_{i12}^* + \varepsilon_i$$

with, X_{ip}^* denoting a standardized value corresponding to:

$X_{i1}, X_{i3}, X_{i4}, X_{i5}, X_{i6}, X_{i7} \sim Discrete\ Uniform\ (0,30)$

$X_{i2}, X_{i8}, X_{i9}, X_{i10}, X_{i11}, X_{i12} \sim Bernoulli\ (.5)$

$\varepsilon_i \sim N(0,1)$

Here X_{i0} equals 1 for all i, therefore β_0 represents the intercept. The remaining variables in the model were independently drawn from the indicated distributions with effects that were zero for all but β_1 and β_2. We consider the situation where $corr(X_{ij}, X_{ij'}) = 0\ for\ j = 1, \dots, 12$, which is motivated by a desire to initially evaluate and the proposed method in the simplest of contexts, a relatively small number of inputs with no correlation.

Data Generation Process 2

Additional simulations considered the more realistic situation where there is a correlation among the inputs $corr(X_{ij}, X_{ij'}) \neq 0$ for some $X_{ij}s$ combined with higher dimensional noise (adding 30 binary variables with null effects), using the following DGP:

$$Y_i = \beta_0 X_{i0} + \beta_1 X_{i1}^* + \cdots + \beta_{42} X_{i42}^* + \varepsilon_i$$

with, X_{ip}^* denoting a standardized value corresponding to:

$X_{i3}, X_{i4}, X_{i5}, X_{i6}, X_{i7} \sim Discrete\ Uniform\ (0,30)$

$X_{i1}, X_{i2}, X_{i8}, X_{i9}, \dots, X_{i42} \sim Bernoulli\ (.5)$

$\varepsilon_i \sim N(0,1)$

with $corr(X_{i2}, X_{i8}) = corr(X_{i1}, X_{i8}) = corr(X_{i1}, X_{i2}) = \rho$ and zero correlation otherwise

Values of ρ were set to 0, .25, .50, or -.33 (this was the maximum negative correlation that could be simulated in this context without obtaining a non-positive definite covariance matrix). We changed $X_{i1} \sim Bernoulli$ (.5) from the previous simulation in order to generate data from a multivariate bernoulli distribution with the desired correlation structure. We also added 30 binary noise variables.

Standardization of variables in both DGP 1 & 2 facilitated assigning values to β_1 and β_2 that were comparable. Parameters β_1 and β_2 were set to be equal and may take on values of .050, .331, 1.134 corresponding to small, medium, or large associations on a correlation scale (r= .05, .30, .60), respectively. The choice of simulating from a discrete uniform distribution is based on a desire to mimic the effect of a continuous random variable. Unfortunately, use of a truly continuous random variable would be computationally too expensive in the context of a simulation study (but not necessarily in an applied context) because it would require generating as many datasets as there are distinct values of the continuous random variable, and then passing each through each tree in the forest. The $X_{i1}, \ldots, X_{i12}$ variables were inputs into the random forest for the DGP-1 and $X_{i1}, \ldots, X_{i42}$ for DGP-2. The effect based on partial dependence was calculated using (6) and confidence intervals were constructed using the nonparametric bootstrap described in (7), as well as a bias-corrected and accelerated version of this interval estimator. We considered 3 effects (.050, .331, 1.134) X 5 sample sizes (40, 100, 250, 500, 1000) x 5 correlation structures/inputs (ρ=0/12inputs, ρ=0/42 inputs, ρ=.25/42 inputs, ρ=.50/42 inputs, ρ=-.33/42 inputs) for point estimation, and for interval estimates we evaluated a more limited number of conditions, in particular excluding conditions with N=1000 given the substantial computational burden.

Computation

The DGPs and fitting of random forests was undertaken in R, the latter based on the randomForest package. The DGP for correlated Xs was based on the MultiOrd package, which implements the method of Emrich and Piedmonte (1991). The default methods in the randomForest package were used for all except one parameter, the number of trees. This was set to 300 because we found little evidence of improvement with increasing number of trees. Nonparametric bootstrapping was implemented using the boot package of R for both the percentile and the bias-corrected and accelerated confidence intervals. The number of bootstrap replicates used for the construction of each interval for each simulated cases was 1000. Given the computationally burdensome design, simulating cases was done in parallel using the snowfall package. Evaluation of point estimation accuracy was executed in parallel on a desktop PC with multi-core processor, whereas larger tasks involving evaluation of interval estimates for any one condition was more problematic (1000 simulated cases x 1000 bootstrap replicates= 1,000,000 random forests) and therefore required use of a computing cluster, the Triton Shared Computing Cluster at the University of California-San Diego.

3.2. Results

3.2.1. DGP 1 Results

Our initial simulations manipulated the number of variables considered at each split. Table 1 displays the results of selecting the number of variables considered at each split (mtry) to be chosen based on a search function (tuneRF) that minimizes the out of bag error for each simulated case. Selection of the number of predictors begins with 4 predictors at each split, and increases or decreases the number of predictors by a factor of 2 (or truncated to the maximum number of predictors-12), stopping when the level of improvement in the out of bag error is less than .01. The results in Table 1 are less than ideal. In Figure 1 we illustrate the effect of manual selection of different numbers of predictors to be used at each split on parameter estimation. The figure conveys a clear improvement with increasing mtry, with the best performance occurring when mtry is set at the maximum. For this reason simulations reported hereafter only considered mtry set at its maximum.

Table 2 displays bias and RMSE as a function of the parameter value and sample size for non-null effects. One result that is immediately clear is that any underestimation bias and overall inaccuracy (as indexed by RMSE) is attenuated by increasing the magnitude of the effect (this decreases the relative bias) and increasing sample size. We have found that the pattern of reduced bias and improved accuracy with increasing effect and sample size corresponds to a greater frequency of these non-null variables appearing in the first two splits of a tree (see Appendix A). Therefore, appearance in early splits appears to be an important determinant of obtaining accurate point estimates. In contrast, for parameters with null effects in the model (e.g., $\beta_3 = \beta_8 = 0$), the bias was calculated to be less than .001 irrespective of the sample size and the magnitude of parameters with non-null effects in the model. As expected, for those variables with null effects the RMSE does decrease with increasing sample size (Table 2).

Based on the coverage reported in Table 3 for non-null effects, we observed improved coverage with increasing effect size and sample size, which is not surprising given the aforementioned improvement in parameter estimation with increasing sample size and effect size. Interestingly, the BCa intervals generally provide superior coverage to the percentile method with medium and large effects while the percentile method appears superior with small and null effects. The difference between the two methods generally decreases with increasing sample size, therefore for larger datasets the decision about which method to use may be less important. In practice, we might prefer BCa intervals when, for instance, there is a notable difference between the mean of the bootstrap estimates and the estimate from the full sample. It is worth noting that we did not find evidence of improvement with an increase in the number of bootstrap replicates. For instance, we used 5,000 bootstrap replicates for the N=40, $\beta = .050$ condition, the coverage with BCa intervals for the non-null effects was .758/.590, which is not substantially different from the Table 3 values of .769/.612.

3.2.2. DGP 2 Results

It does not appear that increasing the number of noise variables alone alters the degree of bias in the estimates (cf. Table 4 for ρ=0 to Table 2 entries). A positive correlation among the inputs leads to less bias in non-null effects than when there is no correlation among the inputs, and this effect increases with increasing positive correlation. In a few limited conditions there is a tendency for positive correlation to lead to an overestimation bias, but the magnitude of the bias is relatively small. In contrast, when there is a negative correlation among the inputs, this leads to increased underestimation bias relative to when there is no correlation for both non-null effects and null effects. We should note for conditions with a negative correlation, the bias decreases with increasing sample size. Collectively, this suggests that higher dimensional noise has little effect on estimation, but the sign of the correlation among inputs can either improve or worsen estimation of this quantity, with improvements generally resulting with increasing sample size.

3.3. Application to Prediction of Abalone Age

The data for this example originates from the Tasmanian Aquaculture and Fisheries Institute, obtainable from the UCI machine learning repository. One way of regulating the harvesting of abalone is through limiting which abalone can be harvested based on their age. Age of abalone can be determined by counting the rings on its shell. Unfortunately, this requires cutting the shell, which would not be an effective way of regulating its harvesting. Therefore, if the age of the abalone can be determined from other physical measurements that would be preferable. The data contained in the dataset aim to achieve this goal. The input variable used in the prediction of the number of rings (+1.5 is a proxy for Abalone age) included: Sex (Infant, Male, Female), Length, Diameter, Height, Whole Weight, Shucked Weight, Viscera Weight, and Shell Weight. We fit a random forest with rings as the response and the variables described above as inputs. We used 300 trees for the random forest and the maximum mtry (8). We used 5000 bootstraps replicates to construct 95% Percentile and BCa confidence intervals. We report on the effects of two variables, sex and length using the proposed method. R programming syntax for this example is provided as supplementary material.

The partial dependence plot for length is provided in Figure 2. The estimate for length was -.562, comparison of infants to females was -.242, and comparison of males to females was .004. There was notable bias in the estimate of length (i.e., estimate of mean from full sample-mean of bootstrap estimates= -.151), suggesting that BCa would be more appropriate. Similar bias was not present for either of the estimates involving comparison of the sex variable (-0.009 (I) & -0.007 (M)). For length the BCa method gives 95% confidence intervals of (-1.325, 0.264) and percentile (-1.671, 0.067). For the infant effect the BCa method gives 95% confidence intervals of (-0.368, -0.035) and percentile (-0.394, -0.077). For the male effect the BCa method gives 95% confidence intervals of (-0.075, 0.094) and percentile (-0.093, 0.079).

4. Discussion

Quantifying a variable's effect is important for the adoption of random forests in future scientific applications. Our simulations suggest estimates are less biased and more accurate with increasing effect and sample size. An important determinant for obtaining unbiased and accurate results appears to be whether variables with true effects appear in early splits. Confidence intervals constructed using a non-parametric bootstrap appears to be an effective method, one that improves with increasing effect and sample size as well.

There are several limitations associated with the simulation study that could lead to fruitful directions for future research. First, we did not consider dichotomous outcome data. One implementation of an approach involving such outcome data could involve calculating partial dependence predictions based on the proportion of events in the terminal nodes of each tree averaged over the trees in the forest, as opposed to predictions based on majority vote, which have been shown in some simulations to be less accurate (Malley et al., 2012). If interest only centers on interpreting binary explanatory variables, it would be straightforward to calculate the risk difference, risk ratio, or odds ratio based on probabilities obtained via partial dependence. For explanatory variables with many response options a feasible alternative would be a weighted beta regression model (Ferrari and Cribari-Neto, 2004) with use of a logit link function leading to an odds ratio interpretation of the exponentiated regression coefficients. Yet another outcome type that is often of interest is time to event, and quantification of a variable's effect could be based on a survival probability calculated using partial dependence at a fixed time point (Ishwaran et al., 2008)

There are alternative modeling approaches that could be considered to achieve the same goals as those considered in this paper. One option that has recently been proposed in the data mining literature is counterfactual machines (Dasgupta et al., 2014). In the simple case of a binary explanatory variable, two forests are built, each using only data from observations belonging to each level of the variable. Predictions for each observation then are based on passing each observation through the forest they were not used to construct, thus providing counterfactual inferences regarding what would have happened if the observation belonged to the other level of the explanatory variable. Although this is an appealing approach, it can present challenges in implementation when explanatory variables are not binary. Yet another approach might involve the use of propensity scores in a conventional parametric model (Rosenbaum and Rubin, 1983). It should be noted however that such approaches rely on correct specification of the propensity score model, which may be difficult to achieve. One possibility is use of a tree-based ensemble method, such as random forests, to estimate each observation's probability of receiving "treatment" (Lee, Lessler, and Stuart, 2010). However, there are important practical limitations with the use of propensity scores. One such problem is that going through the process of using propensity scores for each variable that might be of interest (in terms of the effect that variable has on the outcome) in a research study with a large number of explanatory variables can be quite cumbersome. Therefore, the method proposed in this paper might be considered a reasonable alternative.

Several factors may make the implementation of the methods described in this paper challenging. Larger sample sizes, numbers of inputs, and response options for input variables whose effect will be calculated will increase computing time. In particular, bootstrapping confidence intervals in this context can be a computationally expensive proposition. Fortunately, bootstrapping is a task that is easily parallelized and computation can be sped up by parallelizing on a multi-core processor or a computing cluster. We provide R programming syntax for the empirical example that illustrates how this can easily be accomplished with a multi-core processor. Moreover, increasing computing speeds over time will further increase the feasibility of bootstrapping with large datasets. The method proposed in this paper represents a reasonable approach to quantifying the effect in random forests. The method will allow for a wider adoption of random forest methodology, in particular to applications where the nature and strength of a variable's effect on the outcome is of interest.

References

[1]. Breiman, L. (1996). Bagging Predictors. Machine Learning, 24, 123–140.

[2]. Breiman, L. (2001). Random Forests. *Machine Learning,* 45, 5–32.

[3]. Breiman, L., Friedman, J.H., Olshen, R.A., and Stone, C.J. (1984). *Classification and Regression Trees*. New York: Chapman & Hall.

[4]. Dasgupta, A., Szymczak, S., Moore, J.H., Bailey-Wilson, J.E., and Malley, J.D. (2014). Risk estimation using probability machines. *BioData Mining,* 7, 1-17.

[5]. Efron, B. and Tibshirani, R.J. (1993). *An Introduction to the Bootstrap*. London, U.K.: Chapman & Hall.

[6]. Efron, B. (1987). Better Bootstrap Confidence Intervals. *Journal of the American Statistical Association,* 82, 171-200.

[7]. Emrich, L.J., and Piedmonte, M.R. (1991). A method for generating high-dimensional multivariate binary variates. *American Statistician,* 45, 302-304.

[8]. Ferrari, S.L.P. and Cribari-Neto, F. (2004). Beta regression for modeling rates and proportions. *Journal of Applied Statistics,* 31, 799–815.

[9]. Friedman, J. (2001). Greedy Function Approximation: The Gradient Boosting Machine. *Annals of Statistics,* 29, 1189-1232.

[10]. Hastie,T., Tibshirani, R., and Friedman, J. (2009).*The Elements of Statistical Learning: Prediction, Inference and Data Mining*. New York: Springer Verlag.

[11]. Imbens, G.W. (2004). Nonparametric estimation of average treatment effects under exogeneity: A review. *The Review of Economics and Statistics,* 86, 4–29.

[12]. Ishwaran, H., Kogalur, U.B., Blackstone, E.H., Lauer, M.S. (2008). Random survival forests. *Annals of Applied Statistics,* 2, 841-860.

[13]. Lee, B.K., Lessler, J., and Stuart, E.A. (2010). Improving propensity score weighting using machine learning. *Statistics in Medicine,* 29, 337-346.

[14]. Malley, J.D., Kruppa, J., Dasgupta, A., Malley, K.G., Ziegler, A. **(**2012). Probability machines: consistent probability estimation using nonparametric learning machines. *Methods of Information in Medicine,* 51, 74-81.

[15]. Rosenbaum, P.R., and Rubin, D.B. (1983). The central role of the propensity score in observational studies for causal effects. *Biometrika*, 70, 41–55.

[16]. Strobl, C., Boulesteix, A.L., Kneib, T., Augustin, T., Zeileis A. (2008). Conditional variable importance for random forests. *BMC Bioinformatics,* 9, 307.

[17]. Strobl, C., Malley, J., and Tutz, G. (2009).An Introduction to Recursive Partitioning: Rationale, Application and Characteristics of Classification and Regression Trees, Bagging and Random Forests. *Psychological Methods*, 14, 323–348.

Table 1. Bias and RMSE for DGP 1 with Numerical Search for Optimal *Mtry*

	N=40	N=100	N=250	N=500	N=1000
			Bias		
β =.050	.031/.038	.026/.033	.023/.029	.024/.028	.024/.026
β =.331	.187/.224	.145/.168	.110/.117	.097/.093	.089/.078
β =1.134	.400/.369	.218/.084	.094/.034	.076/.0267	.068/.023
			RMSE		
β =.050	.083/.060	.067/.050	.044/.041	.036/.036	.031/.031
β =.331	.217/.242	.174/.191	.135/.141	.118/.113	.103/.093
β =1.134	.449/.467	.262/.169	.130/.091	.106/.067	.092/.050

Note: Bias=parameter-estimate. The value to the left of the slash corresponds to the discrete uniform random variable and to the right is the Bernoulli random variable.

Table 2. Bias and RMSE for DGP 1 with Maximum *Mtry*

	N=40	N=100	N=250	N=500	N=1000
			Bias (Non-Null Effects)		
β =.050	.019/.038	.011/.033	.008/.028	.005/.025	.004/.020
β =.331	.111/.208	.062/.132	.027/.053	.018/.015	.011/.002
β =1.134	.321/.328	.164/.039	.041/.004	.024/.000	.014/.000
			RMSE (Non-Null Effects)		
β =.050	.123/.064	.099/.053	.057/.043	.043/.038	.030/.032
β =.331	.188/.245	.129/.182	.073/.105	.050/.059	.034/.034
β =1.134	.377/.443	.209/.136	.079/.069	.053/.047	.036/.033
			RMSE (Null Effects)		
β =.050	.122/.049	.098/.038	.055/.029	.041/.022	.030/.017
β =.331	.115/.045	.084/.033	.050/.023	.038/.019	.028/.014
β =1.134	.081/.037	.055/.024	.043/.019	.034/.016	.026/.013

Note: Bias=parameter-estimate. The value to the left of the slash corresponds to the discrete uniform random variable and to the right is the Bernoulli random variable. Bias for null effects is <.001 in all simulated conditions of this design

Table 3. Coverage for Non-Null and Null Effects from DGP 1 with Maximum *Mtry*

	N=40	N=100	N=250	N=500
Non-Null Effects				
BC$_a$				
β =.050	.769/.612	.796/.637	.869/.662	.879/.742
β =.331	.832/.737	.868/.886	.911/.918	.899/.955
β =1.134	.739/.944	.762/.966	.908/.911	.898/.904
Percentile				
β =.050	.905/.629	.902/.657	.919/.645	.920/.697
β =.331	.749/.450	.831/.636	.865/.792	.873/.890
β =1.134	.471/.644	.567/.876	.835/.914	.858/.912
Null Effects				
BC$_a$				
β =.050	.766/.722	.809/.760	.860/.760	.858/.780
β =.331	.783/.743	.859/.764	.875/.778	.889/.833
β =1.134	.893/.817	.845/.830	.856/.840	.861/.848
Percentile				
β =.050	.950/.951	.939/.939	.917/.929	.919/.921
β =.331	.939/.966	.950/.953	.934/.925	.940/.933
β =1.134	.975/.975	.955/.985	.950/.970	.928/.953

Note: The value to the left of the slash for non-null effects corresponds to the discrete uniform random variable and to the right the Bernoulli random variable.

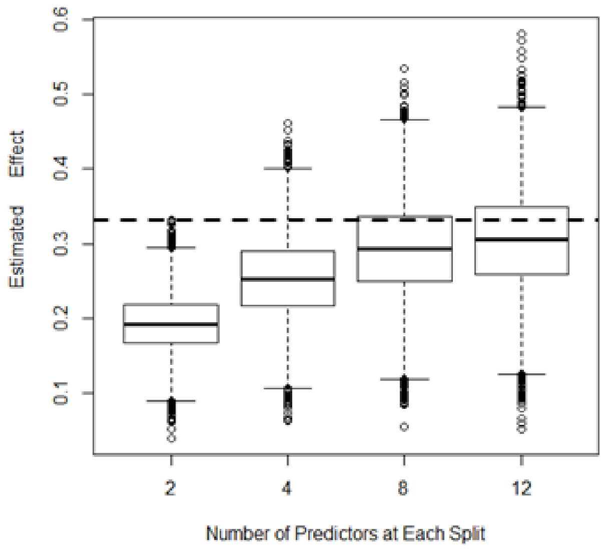
Table 4. Bias for Null and Non-Null Effects for DGP 2 with Maximum *Mtry*

	N=40	N=100	N=250	N=500	N=1000
			ρ=0		
Non-Null Effects					
β =.050	.042/.042	.038/.038	.034/.034	.030/.030	.027/.026
β =.331	.232/.232	.150/.153	.064/.064	.019/.018	.000/.000
β =1.134	.266/.268	.006/.005	.000/-.001	.000/.000	.000/.000
Null Effects					
β =.050	.000/.000	.000/.000	.000/.000	.000/.000	.000/.000
β =.331	.000/.000	.000/.000	.000/.000	.000/.000	.000/.000
β =1.134	.000/.000	.000/.000	.000/.000	.000/.000	.000/.000
			ρ=.25		
Non-Null Effects					
β =.050	.040/.040	.036/.036	.032/.031	.027/.026	.022/.022
β =.331	.194/.191	.105/.104	.050/.048	.016/.018	.000/.002
β =1.134	.161/.162	.005/.008	.001/-.001	.000/.000	.001/.000
Null Effects					
β =.050	.000/-.003	.000/-.005	.000/-.005	.000/-.006	.000/-.005
β =.331	.000/-.018	.000/-.014	.000/-.008	.000/-.003	.000/.000
β =1.134	.001/-.016	.000/-.001	.001/.000	-.001/.000	.000/.000
			ρ=.50		
Non-Null Effects					
β =.050	.040/.039	.034/.034	.029/.029	.025/.024	.019/.019
β =.331	.156/.156	.071/.070	.042/.039	.020/.019	.002/.004
β =1.134	.130/.132	.020/.010	-.001/.001	.001/.000	.000/.000
Null Effects					
β =.050	.000/-.006	.001/-.008	.000/-.009	.000/-.009	.000/-.001
β =.331	.000/-.040	.001/-.027	.000/-.016	.000/-.009	.000/-.002
β =1.134	.000/-.045	.001/-.007	.000/.000	-.001/.000	.000/.000
			ρ= -.33		
Non-Null Effects					
β =.050	.045/.044	.042/.042	.039/.040	.037/.037	.035/.035
β =.331	.283/.283	.241/.241	.160/.159	.086/.086	.040/.040
β =1.134	.686/.685	.221/.223	.091/.092	.070/.071	.055/.056
Null Effects					
β =.050	-.001/.004	.002/.006	.000/.009	.000/.010	.000/.011

β =.331	.000/.039	.000/.059	.000/.067	.000/.053	.000/.034
β =1.134	.000/.188	-.001/.157	.000/.094	.000/.076	.000/.058

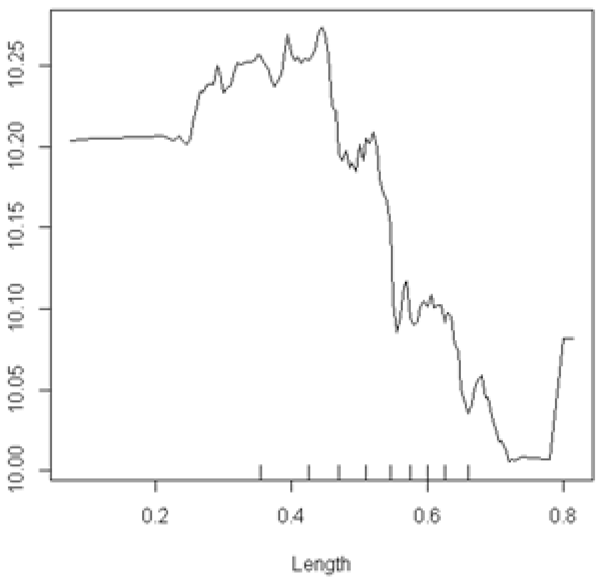
Note: Bias=parameter-estimate. The values to the left and right of the slash both correspond to Bernoulli random variables for non-null effects. The value to the left of the slash in the null case corresponds to a discrete uniform random variable and to the right a Bernoulli random variable.

Figure 1. Effect of *Mtry* on Parameter Estimation



Note: For each simulated case N=250. Dashed line is the parameter $\beta = .331$

Figure 2. Partial Dependence on Length



Appendix A

Here we considered the effect of variable splits on increasing effect size, sample size, and mtry. To examine this issue in greater depth we examined the nature of variable splits that take place in the simulated conditions. We focused on two effects (β =.050 vs. 1.134), two sample sizes (N=40 vs. 1000), and two values of mtry (2 vs. 12). We considered whether or not a variable was used in the first two splits of a tree. These results were averaged over all trees in the forest and over all simulated cases. Table A1 displays the result.

Table A1. Proportion of Times Variables Appear in First Two Splits

		X_1		X_3	
	mtry	N=40	N=1000	N=40	N=1000
β =1.134	2	0.271	0.286	0.187	0.192
	12	0.853	1.000	0.076	0.000
β =.050	2	0.212	0.241	0.211	0.223
	12	0.253	0.415	0.250	0.259
		X_2		X_8	
	mtry	N=40	N=1000	N=40	N=1000
β =1.134	2	0.242	0.302	0.082	0.078
	12	0.610	1.000	0.013	0.000
β =.050	2	0.095	0.131	0.093	0.090
	12	0.043	0.099	0.038	0.027

Note: $X_1$is a discrete uniform random variable with a population effect>0 and X_3 with a population effect=0. X_2 is a Bernoulli random variable with a population effect>0 and X_8 with a population effect=0.

We can see in all conditions that the variables with the effect (X_1 & X_2) appear more often in the first two splits of a tree than the variables without the effect (X_3 & X_8). This is much more pronounced in the large effect conditions, especially those with larger mtry. Increasing sample size increases the appearance of a variable in the first two splits if the variable has any effect, otherwise it decreases a variable's appearance. The appearance of variables in early splits parallels the general pattern of results related to bias and RMSE.

When a variable has a relationship to the response, increasing mtry, effect size, and sample size will increase a variable’s appearance in the first two splits of a tree. Therefore, it seems that appearance in early splits is an important determinant of obtaining accurate and unbiased point estimates. One explanation for the effect of mtry is that in random forests with mtry less than the maximum, non-null variables will be omitted in some early tree splits simply due to random feature selection. This will lead to a systematic attenuation of the effect of those variables, leading to bias and inaccuracy. However, this cannot happen in bagging (i.e., when mtry is the maximum) because all variables have a chance to appear in all tree splits, most importantly the early ones, leading to less bias and inaccuracy. While all variables are considered at early splits in bagging, some variables with null population effects may still be chosen just by chance. When the effect sizes for the variables that have non-null population effects are small, it is increasingly likely that variables with null population effects will be chosen in their place, which in turn attenuates the estimate of the effect and increases bias and inaccuracy. However, with increasing effect size the chances that variables with null population effects are selected decreases, limiting the bias and inaccuracy. Lastly, when the sample sizes are small the “true” effect of a variable will be less reliable, decreasing the chances that it will be chosen when it has a non-null population effect, a situation that improves with increasing sample size.

SUPPLEMENTARY MATERIALS

```
# Steps to implementing Abalone example: 1) Load library 2) create new partialPlot function called
partialPlot.mod

# Load necessary libraries
library(randomForest)
library(boot)

#Creating a modified partialPlot function called partialPlot.mod (uses distinct X values in data to
construct partial dependence for non-factor #variables)
partialPlot.mod<-function (x, pred.data, x.var, which.class, w, plot = TRUE, add = FALSE,
  n.pt = min(length(unique(pred.data[, xname])), 51), rug = TRUE,
  xlab = deparse(substitute(x.var)), ylab = "", main = paste("Partial Dependence on",
    deparse(substitute(x.var))), ...)
{
  classRF <- x$type != "regression"
  if (is.null(x$forest))
    stop("The randomForest object must contain the forest.\n")
  x.var <- substitute(x.var)
  xname <- if (is.character(x.var))
    x.var
  else {
    if (is.name(x.var))
      deparse(x.var)
    else {
      eval(x.var)
    }
  }
  xv <- pred.data[, xname]
  n <- nrow(pred.data)
  if (missing(w))
    w <- rep(1, n)
  if (classRF) {
    if (missing(which.class)) {
      focus <- 1
    }
    else {
      focus <- charmatch(which.class, colnames(x$votes))
      if (is.na(focus))
        stop(which.class, "is not one of the class labels.")
    }
  }
```

```
if (is.factor(xv) && !is.ordered(xv)) {
   x.pt <- levels(xv)
   y.pt <- numeric(length(x.pt))
   for (i in seq(along = x.pt)) {
      x.data <- pred.data
      x.data[, xname] <- factor(rep(x.pt[i], n), levels = x.pt)
      if (classRF) {
         pr <- predict(x, x.data, type = "prob")
         y.pt[i] <- weighted.mean(log(ifelse(pr[, focus] >
           0, pr[, focus], .Machine$double.eps)) - rowMeans(log(ifelse(pr >
           0, pr, .Machine$double.eps))), w, na.rm = TRUE)
      }
      else y.pt[i] <- weighted.mean(predict(x, x.data),
         w, na.rm = TRUE)
   }
   if (add) {
      points(1:length(x.pt), y.pt, type = "h", lwd = 2,
         ...)
   }
   else {
      if (plot)
         barplot(y.pt, width = rep(1, length(y.pt)), col = "blue",
           xlab = xlab, ylab = ylab, main = main, names.arg = x.pt,
           ...)
   }
}
else {
   if (is.ordered(xv))
      xv <- as.numeric(xv)
   x.pt <- sort(unique(xv))
   y.pt <- numeric(length(x.pt))
   for (i in seq(along = x.pt)) {
      x.data <- pred.data
      x.data[, xname] <- rep(x.pt[i], n)
      if (classRF) {
         pr <- predict(x, x.data, type = "prob")
         y.pt[i] <- weighted.mean(log(ifelse(pr[, focus] ==
           0, .Machine$double.eps, pr[, focus])) - rowMeans(log(ifelse(pr ==
           0, .Machine$double.eps, pr))), w, na.rm = TRUE)
      }
      else {
         y.pt[i] <- weighted.mean(predict(x, x.data),
           w, na.rm = TRUE)
```

```
            }
        }
        if (add) {
            lines(x.pt, y.pt, ...)
        }
        else {
            if (plot)
                plot(x.pt, y.pt, type = "l", xlab = xlab, ylab = ylab,
                  main = main, ...)
        }
        if (rug && plot) {
            if (n.pt > 10) {
                rug(quantile(xv, seq(0.1, 0.9, by = 0.1)), side = 1)
            }
            else {
                rug(unique(xv, side = 1))
            }
        }
    }
    invisible(list(x = x.pt, y = y.pt))
}

# Data available from UCI machine learning repository @
http://archive.ics.uci.edu/ml/datasets/Abalone
abalone <- read.table("C:/Documents and Settings/M931496/Desktop/abalone.data.dat",sep=",")
abaloneNames <-
c("Sex","Length","Diameter","Height","Whole.weight","Shucked.weight","Viscera.weight","Shell.w
eight","Rings")
colnames(abalone) <- abaloneNames

rf<-randomForest(Rings~., data=abalone, mtry=8, ntree=300)

# Calculating point estimates

#Sex
predictdep.Sex<-partialPlot(x=rf, pred.dat=abalone, x.var=Sex, plot=TRUE)
lm(y~x, data=predictdep.Sex, weights=table(abalone$Sex))

#Length
predictdep.Length<-partialPlot.mod(x=rf, pred.dat=abalone, x.var=Length, plot=TRUE)
lm(y~x, data=predictdep.Length, weights=table(abalone$Length))
```

```
#Bootstrapping CI

bs<-function(d,z){
d2<-abalone[z,]
rf<-randomForest(Rings~., data=d2, mtry=8, ntree=300)

#Sex
predictdep.Sex<-partialPlot(x=rf, pred.dat=abalone, x.var=Sex, plot=FALSE)
marg.model.Sex<-lm(y~x, data=predictdep.Sex, weights=table(abalone$Sex))
beta.Sex<-data.frame(marg.model.Sex$coefficients)
marg.effect.Sex1<-beta.Sex[2,]
marg.effect.Sex2<-beta.Sex[3,]


#Length
predictdep.Length<-partialPlot.mod(x=rf, pred.dat=abalone, x.var=Length, plot=FALSE)
marg.model.Length<-lm(y~x, data=predictdep.Length, weights=table(abalone$Length))
beta.Length<-data.frame(marg.model.Length$coefficients)
marg.effect.Length<-beta.Length[2,]

all<-rbind(marg.effect.Length, marg.effect.Sex1, marg.effect.Sex2)
return(all)

}

#This can take a while when using a single core with a large number of bootstrap replicates with
the Abalone dataset
# To reduce time consider parallel processing (see further below for implementation)

results<-boot(data= abalone, statistic=bs, R=5000)

boot.ci(results, type="bca", index=)
boot.ci(results, type="bca", index=1)
boot.ci(results, type="perc", index=1)
boot.ci(results, type="bca", index=2)
boot.ci(results, type="perc", index=2)
boot.ci(results, type="bca", index=3)
boot.ci(results, type="perc", index=3)


#Parallel Computing
# Load some more libraries
```

```
#Using parallel library just to get number of cores on my computer
library(parallel)
detectCores()

library(snow)
library( rlecuyer )

#detectCores tells me I have 4 cores so making a cluster of 4
cl <- makeCluster(4)
#Setting up independent random number generation streams on each core
clusterSetupRNG(cl, seed=245)
#confirming independence of streams
clusterCall(cl, runif, 4)
#Exporting dataset to each core
clusterExport(cl, "abalone")
#Loading libraries on each core
clusterEvalQ(cl, library(boot))
clusterEvalQ(cl, library(randomForest))
#Loading the new function on each core
clusterEvalQ(cl,
partialPlot.mod<-function (x, pred.data, x.var, which.class, w, plot = TRUE, add = FALSE,
    n.pt = min(length(unique(pred.data[, xname])), 51), rug = TRUE,
    xlab = deparse(substitute(x.var)), ylab = "", main = paste("Partial Dependence on",
        deparse(substitute(x.var))), ...)
{
    classRF <- x$type != "regression"
    if (is.null(x$forest))
        stop("The randomForest object must contain the forest.\n")
    x.var <- substitute(x.var)
    xname <- if (is.character(x.var))
        x.var
    else {
        if (is.name(x.var))
            deparse(x.var)
        else {
            eval(x.var)
        }
    }
    xv <- pred.data[, xname]
    n <- nrow(pred.data)
    if (missing(w))
        w <- rep(1, n)
```

```
if (classRF) {
   if (missing(which.class)) {
      focus <- 1
   }
   else {
      focus <- charmatch(which.class, colnames(x$votes))
      if (is.na(focus))
         stop(which.class, "is not one of the class labels.")
   }
}
if (is.factor(xv) && !is.ordered(xv)) {
   x.pt <- levels(xv)
   y.pt <- numeric(length(x.pt))
   for (i in seq(along = x.pt)) {
      x.data <- pred.data
      x.data[, xname] <- factor(rep(x.pt[i], n), levels = x.pt)
      if (classRF) {
         pr <- predict(x, x.data, type = "prob")
         y.pt[i] <- weighted.mean(log(ifelse(pr[, focus] >
           0, pr[, focus], .Machine$double.eps)) - rowMeans(log(ifelse(pr >
           0, pr, .Machine$double.eps))), w, na.rm = TRUE)
      }
      else y.pt[i] <- weighted.mean(predict(x, x.data),
         w, na.rm = TRUE)
   }
   if (add) {
      points(1:length(x.pt), y.pt, type = "h", lwd = 2,
         ...)
   }
   else {
      if (plot)
         barplot(y.pt, width = rep(1, length(y.pt)), col = "blue",
           xlab = xlab, ylab = ylab, main = main, names.arg = x.pt,
           ...)
   }
}
else {
   if (is.ordered(xv))
      xv <- as.numeric(xv)
   x.pt <- sort(unique(xv))
   y.pt <- numeric(length(x.pt))
   for (i in seq(along = x.pt)) {
      x.data <- pred.data
      x.data[, xname] <- rep(x.pt[i], n)
      if (classRF) {
        pr <- predict(x, x.data, type = "prob")
        y.pt[i] <- weighted.mean(log(ifelse(pr[, focus] ==
          0, .Machine$double.eps, pr[, focus])) - rowMeans(log(ifelse(pr ==
          0, .Machine$double.eps, pr))), w, na.rm = TRUE)
      }
      else {
        y.pt[i] <- weighted.mean(predict(x, x.data),
          w, na.rm = TRUE)
      }
    }
    if (add) {
      lines(x.pt, y.pt, ...)
    }
    else {
      if (plot)
        plot(x.pt, y.pt, type = "l", xlab = xlab, ylab = ylab,
          main = main, ...)
    }
    if (rug && plot) {
      if (n.pt > 10) {
        rug(quantile(xv, seq(0.1, 0.9, by = 0.1)), side = 1)
      }
      else {
        rug(unique(xv, side = 1))
      }
    }
  }
  invisible(list(x = x.pt, y = y.pt))
}
)

#Doing the bootstrap
Result<-boot(data= abalone, statistic=bs, R=5000, parallel= "snow", ncpus=4, cl=cl)

boot.ci(results, type="bca", index=)
boot.ci(results, type="bca", index=1)
boot.ci(results, type="perc", index=1)
boot.ci(results, type="bca", index=2)
boot.ci(results, type="perc", index=2)
boot.ci(results, type="bca", index=3)
boot.ci(results, type="perc", index=3)
```

Received December 15, 2015; accepted January 10, 2016.

Barbara A. Bailey, PhD
San Diego State University
bbailey@mail.sdsu.edu
Correspondence:
Guy Cafri, PhD
Child & Adolescent Services Research Center
3665 Kearny Villa Road, Ste 200N
San Diego, CA 92123
Phone: (813) 486-9875
Fax: (858) 408-3850
Email: guycafri@gmail.com