

Running head: BINARY RECURSIVE PARTITIONING

Binary recursive partitioning: Background, methods, and application to psychology

Edgar C. Merkle      Victoria A. Shaffer

Department of Psychology

Wichita State University

**Abstract**

Binary recursive partitioning (BRP) is a computationally-intensive statistical method that can be used in situations where linear models are often used. Instead of imposing many assumptions to arrive at a tractable statistical model, BRP simply seeks to accurately predict a response variable based on values of predictor variables. The method outputs a decision tree depicting the predictor variables that were related to the response variable, along with the nature of the variables' relationships. No significance tests are involved, and the tree's "goodness" is judged based on its predictive accuracy. In this paper, we describe BRP methods in a detailed manner and illustrate their use in psychological research. We also provide R code for carrying out the methods.

## Binary recursive partitioning: Background, methods, and application to psychology

Binary recursive partitioning (BRP), also referred to as *Classification and Regression Trees* (CART)<sup>1</sup>, is a computationally-intensive statistical method that is often attributed to Breiman, Friedman, Olshen, and Stone (1984). The method can be used in situations where one is interested in studying the relationships between a response variable and predictor variables, with “classification” referring to trees with categorical response and “regression” referring to trees with continuous response. The method has many desirable properties, some of the most notable being: (1) it is nonparametric (in the sense that no stochastic model is imposed on the data) and free of significance tests; (2) the predictor and response variables can be of all types (continuous, ordinal, categorical), with minimal change in the underlying algorithm and the resulting output; (3) missing data are handled without the need for imputation techniques; (4) it is invariant to monotonic transformations of the predictor variables; and (5) it is minimally impacted by outliers. As discussed later in the paper, these attributes prove advantageous in some situations where linear models are suboptimal or where the data are exceedingly “messy.”

Modern BRP methods, including those developed by Breiman et al. (1984), are related to earlier procedures that supplement regression analyses. Many of these early procedures were developed in the social sciences (specifically, at the Institute for Social Research, University of Michigan). The most well-known procedure may be the *Automatic Interaction Detection* method developed by Morgan and Sonquist (1963; see also Sonquist, 1970). Automatic Interaction Detection (AID) was proposed as a way of identifying complex interactions in datasets with continuous response variables. The technique involves sequentially splitting the data into two groups based on values of predictor variables, such that the between-groups sum of squares is maximized at each split. A

specific split is obtained by calculating the between-groups sum of squares for each possible split and choosing the split that yields the largest sum of squares.

The original AID procedure was extended to handle categorical response variables (Morgan & Messenger, 1973) and to improve user accessibility (Sonquist, Baker, & Morgan, 1974), and similar procedures were developed for ordinal response variables (e.g., Bouroche & Tennenhaus, 1971). Fielding (1977) presents an excellent overview of these early procedures and their use in practice. A general difficulty was that the results were unstable. For example, Sonquist et al. (1974) state: “A warning to potential users of this program: data sets with a thousand cases or more are necessary; otherwise the power of the search processes must be restricted drastically or those processes will carry one into a never-never land of idiosyncratic results” (p. 3). Modern BRP methods employ a variety of strategies to make results more stable, or at least to give the user more information about the procedures’ general predictive abilities.

While there is a body of research devoted to modern BRP (Ripley, 1996; Zhang & Singer, 1999 present detailed reviews), the literature is largely either: (1) very technical, or (2) application-based and glossing over details of the procedure. Furthermore, while there have been some recent applications and development of BRP in psychology (e.g., Dusseldorp & Meulman, 2004), we have found BRP to be relatively unknown in the field. This is unfortunate because the procedure has roots in the social sciences and could be useful to many research endeavors. BRP is also a “gateway method,” in the sense that variants of BRP drive more advanced methods that are excellent tools for prediction. These advanced methods are briefly described at the end of the tutorial.

The purpose of this paper is to provide an overview of BRP, including information about each step of the procedure and about carrying out the procedure in R. In the following pages, we first provide an introductory example and then discuss specific details about BRP. The description draws from Breiman et al. (1984), as well as from our own

experiences with coding a BRP program from scratch.<sup>2</sup> Next, we describe the application of BRP to a psychology study involving the decisions of mock jurors in malpractice lawsuits. Finally, we compare BRP with regression and discuss some general information regarding the use of BRP in practice.

## BRP Methods

The term *binary recursive partitioning* describes three crucial parts of the underlying algorithm: *partitioning* describes the fact that the algorithm predicts a response variable by partitioning the data into subgroups based on the predictor variables; *binary* describes the fact that, at any one step, the algorithm partitions the data into two subgroups; and *recursive* describes the fact that, within the subgroups created from one predictor variable, the algorithm goes on to partition the data based on other predictor variables or other splits of the same predictor. BRP methods commonly assume independent and identically-distributed observations, but they make no assumptions about error distributions or linearity (Breiman et al., 1984).

Imagine deciding which of a set of predictor variables are important for predicting housing prices in Boston. An intuitive strategy for accomplishing this task is to search for splits in predictor variables that seem predictive of housing prices. For example, one might try to split houses into two groups based on number of rooms, with the “fewer rooms” group predicted to be less expensive than the “more rooms” group. Within these two “number of rooms” groups, one might further split houses into subgroups based on, say, distances from each house to the city. This is the essence of BRP: it is an algorithm that searches through all predictor variables to find the split that is most predictive of the response variable. After this split is established, the algorithm continues to search for subsplits that further improve predictions on the response variable. The algorithm generally stops when the data have been split into very small subgroups. After the

algorithm stops, the results may be examined to determine which splits are useful for predicting future data.

Prediction of future data is important because we are usually interested in the extent to which statistical results generalize to new data. By adding parameters (or, for BRP, splits) to a model, we can always improve our ability to predict an observed dataset. However, these extra parameters (splits) often reduce the model's ability to predict future, unseen data. These ideas underlie *generalizability* measures, which attempt to assess the extent to which a fitted model can predict future data. As described in detail below, BRP relies on a generalizability criterion (specifically employing cross-validation methods) to determine the optimal number of splits in a tree. This is generally consistent with other statistical learning algorithms, where heavy emphasis is placed on prediction and little emphasis is placed on parameterized, stochastic models (see, e.g., Breiman, 2001b; Hastie, Tibshirani, & Friedman, 2001).

Returning to the example, a decision tree for predicting housing prices is presented in Figure 1. This is standard BRP output. For a given observation, we start at the top of the tree and follow different branches depending on conditions involving the predictor variables. Trees with multiple layers of splits may be conceptualized as describing interactions between predictor variables. Once we arrive at an endpoint of the tree, we obtain a prediction for the observation. Each circle or rectangle in the tree is called a *node*, and each rectangle in the tree is specifically called a *terminal node*. Finally, the branches that come out of a node are called *splits*.

The specific data on which Figure 1 is based involve 13 variables that are potentially predictive of the median value of homes within different neighborhoods around Boston (Belsley, Kuh, & Welsch, 1980; Harrison & Rubenfeld, 1978).<sup>3</sup> These variables were measured on 506 neighborhoods, and they include such attributes as “per capita crime rate,” “proportion of non-retail business acreage,” and “average number of rooms per

house.” Binary recursive partitioning is used to select the variables that are most predictive of median home value, as well as to discern the nature of the relationship between selected predictor variables and median home value.

The utility of the tree is that, starting with 13 predictor variables, we have selected 4 that are most useful for predicting housing prices: the average number of rooms in a neighborhood (`Room`), the percentage of lower-class individuals in a neighborhood (`Low.class`), the crime rate (`Crime`), and the distance from the house to major employment centers (`Dist`). The tree shows the nature of the predictor variables’ relationships with the response variable. Furthermore, this specific tree was selected as the one that best generalizes to future data. We will interpret this tree in more detail below.

In the sections below, we formally describe the methods underlying BRP. These sections are divided into three parts corresponding to the different steps involved in BRP analysis. First, we address how the splits in the tree are established. Second, we describe how to decide which splits generalize to the prediction of new data. Finally, we describe the interpretation of BRP tree diagrams. The technical part of the discussion below will focus on a situation where the response variable is categorical; the handling of continuous response variables in BRP is very similar. Technical detail for BRP with continuous response variables appears in Appendix A.

### *Creating the Tree*

Trees are built by finding the split on a predictor variable ( $\mathbf{x}_i$ ) that best discriminates between classes of the categorical response variable  $\mathbf{y}$ . There are multiple mathematical definitions of “best” in the preceding sentence, and, for trees, these definitions are generally called *impurity functions*. Splits that do a good job of discriminating between classes of  $\mathbf{y}$  result in nodes that have low impurity, and splits that do not discriminate between classes of  $\mathbf{y}$  result in nodes that have high impurity. A

popular impurity function in BRP is the *Gini criterion*; it can be written as:

$$i(t) = 1 - \sum_j p^2(j | t), \quad (1)$$

where  $p(j | t)$  is the probability that an observation is actually in class  $j$  given that it falls in node  $t$ . The Gini criterion is 0 when only class  $j$  observations fall in a node (perfect discrimination within the node), and it is at a maximum when each class falls in a node with equal probability. For a response variable  $\mathbf{y}$  with two classes, the Gini criterion is plotted as a function of  $p(j | t)$  in Figure 2. If half of a node's observations fall in each of two classes, then  $p(\text{class 1} | t) = p(\text{class 2} | t) = .5$ . Thus, as shown in Figure 2 and as defined by the Gini criterion, the node impurity equals  $1 - (.5^2 + .5^2) = .5$ . Conversely, when all of a node's observations fall in one class,  $p(\text{class 1} | t) = 1$  and  $p(\text{class 2} | t) = 0$  (or vice versa). In this case, the node impurity obtains a minimum of  $1 - (1^2 + 0^2) = 0$ .

The Gini criterion is used to determine optimal splits in the following manner. Starting with some node  $t$ , we seek a new split  $s$  that forms two new nodes,  $t_L$  and  $t_R$ . The  $s$  that we choose will be the one that leads to the greatest decrease in the Gini criterion. We can write this decrease as:

$$\Delta i(s, t) = i(t) - p_L i(t_L) - p_R i(t_R), \quad (2)$$

where  $p_L$  is the empirical probability that an observation is split into node  $t_L$  and  $p_R = 1 - p_L$  is the empirical probability than an observation is split into node  $t_R$ . Predictions for the new nodes,  $t_L$  and  $t_R$ , correspond to the predominant  $\mathbf{y}$  category within the nodes. For example, if 80% of the observations in node  $t_L$  are class 1 observations (and 20% are class 2 observations), “class 1” would be the prediction for all observations in node  $t_L$ .

For a categorical  $\mathbf{x}_i$  with  $k$  levels, there are  $2^{(k-1)} - 1$  possible splits of those levels

into two groups. Each of the  $2^{(k-1)} - 1$  possible splits is considered by calculating the extent to which the node impurity, as defined by the Gini criterion, is reduced. For continuous  $\mathbf{x}$ 's, we can recognize that, once data have been observed, the data may be treated as ordinal. That is, if we have 100 observations on a continuous variable  $\mathbf{x}_i$ , then there are 99 possible ways in which we could split those observations into two groups.<sup>4</sup> We could put the lowest of the  $\mathbf{x}_i$ 's in one group and the ninety-nine higher  $\mathbf{x}_i$ 's in another group; we could put the two lowest  $\mathbf{x}_i$ 's in one group and the ninety-eight higher  $\mathbf{x}_i$ 's in another group; . . . ; we could put the ninety-nine lowest  $\mathbf{x}_i$ 's in one group and the highest  $\mathbf{x}_i$  in another group. In this way, all possible splits are searched across all  $\mathbf{x}$ 's to find the split that maximizes Equation 2. Then, within the split that was just established, the algorithm starts over and searches for new splits that maximize Equation 2. This process continues until a (liberal) pre-defined stopping point is reached. For example, we might let the tree continue to grow until no fewer than 10 observations fall in a terminal node. Alternatively, we might let the tree grow until there exists no split that improves predictions by a certain amount.<sup>5</sup>

It is very difficult to stop the tree-building process at an optimal point; that is, at a point where the tree best predicts future data. The stopping rules are thus intended to yield trees that overfit the data; that is, to build trees that fit the current data well but that do a poor job of predicting future data. To improve the tree's ability to predict future data, we can attempt to remove splits that do not contribute much to the tree's predictive accuracy. In reference to the fact that we are "cutting back" the tree, this procedure is called *pruning*.

### *Pruning the Tree*

Tree pruning can be characterized as a sequential deletion of uninfluential splits in a tree, resulting in a set of trees that are all nested within the original tree. A final tree is

then selected from this set of trees by considering each tree's ability to predict new data. The following steps are described individually below: (1) deciding which part of the tree to prune; and (2) measuring each tree's ability to predict new data.

### *Pruning Rules*

In tree pruning, we generally start with a large tree  $T$ . We want to snip off the branch of  $T$  that contributes least to  $T$ 's predictive accuracy. For example, referring to Figure 1, it may be the case that the split in the lower left-hand corner (`crim>=7/crim<7`) contributes least to the tree's predictive accuracy. We would delete this split from the tree, resulting in a new tree that is nested within the original tree.

While the Gini criterion is used in the tree-building algorithm to determine splits, misclassification rates are used in the pruning algorithm. Specifically, to find the least important branch of  $T$ , the pruning algorithm utilizes the following equation:

$$G(t) = \frac{R(T_{-t}) - R(T)}{d} \quad (3)$$

where  $T_{-t}$  is a subtree of  $T$  with the branches coming from node  $t$  removed;  $R(T_{-t})$  is the misclassification rate of subtree  $T_{-t}$ ;  $R(T)$  is the misclassification rate of the full tree  $T$ ; and  $d$  is the difference between the number of terminal nodes in  $T$  and the number of terminal nodes in  $T_{-t}$ . This equation measures the extent to which the misclassification rate decreases from tree  $T_{-t}$  to tree  $T$ . In dividing by  $d$ , the equation specifically measures decrease in misclassification rate per extra terminal node in  $T$ .

Equation (3) is always greater than or equal to 0, because a subtree  $T_{-t}$  can never make more accurate predictions than the full tree  $T$ . When Equation (3) is close to 0, the extra splits in tree  $T$  are not very useful: the misclassification rate does not change much when considering the extra terminal nodes. When Equation (3) is further from 0, the extra terminal nodes in tree  $T$  are worthwhile: the misclassification rate considerably

decreases from  $T_{-t}$  to  $T$ , even after accounting for the extra terminal nodes in  $T$ .

The pruning algorithm specifically utilizes Equation (3) as follows:

1. Start with the full tree  $T$ .
2. For all non-terminal nodes in  $T$ , calculate  $G(t)$  using Equation (3).
3. Find the non-terminal node yielding the smallest value of  $G(t)$ . Remove all branches stemming from that node, yielding a new tree  $T_{-t}$ .
4. Repeat steps 1-4, using the new tree  $T_{-t}$  in place of  $T$ .

The algorithm continues in this manner until no more splits remain in the tree. The smallest value of  $G(t)$  at each step of the algorithm is sometimes called the “complexity parameter.”

As an example, consider the hypothetical scenario in Table 1, which lists the misclassification rates and number of terminal nodes for a full tree (labeled T) and four subtrees (labeled 1 – 4). Equation (3) is used to calculate  $G(t)$  for each subtree; for example, for subtree 1,  $G(t) = (.63 - .6)/(10 - 9) = .030$ . This ends up being the smallest value of  $G(t)$  among the four subtrees, so we would prune the full tree to subtree 1 and repeat the steps listed above (with subtree 1 taking the place of T).

To summarize, we start with a full tree  $T$  and use the above algorithm to commit a series of prunings on  $T$ . This gives us a set of trees, each of which have a different number of terminal nodes and are nested within  $T$ . As described below, we then use cross-validation to examine each tree’s ability to predict new data. We choose the tree that yields the most accurate predictions on these new data.

### *Choosing the Best Tree*

Of the set of trees obtained from the pruning algorithm, which would best predict new data? We usually do not have an entirely new dataset on which we can test each tree,

so we rely on *k-fold cross-validation* to examine BRP's predictions on new data. *k*-fold cross-validation (applied to BRP by Breiman et al., 1984; see Browne, 2000 for a general presentation) involves splitting up one's current dataset to build a tree with one part (the “training” set) and to examine tree predictions with the other part (the “validation” set).

To be more specific, the original dataset is split into  $k$  subsets, and each subset is sequentially held out as a tree is being built and pruned. Thus, we end up with  $k$  sets of pruned trees:  $T^{(1)}, T^{(2)}, \dots, T^{(k)}$ . While these  $k$  sets of trees will be different from one another (in terms of splits and predictor variables used), they can be approximately matched to one another in terms of tree size (as measured by number of terminal nodes).<sup>6</sup> Thus, for a given tree size, we can examine the validation set predictions across the  $k$  sets of trees. We choose the “best-sized” tree as that which yields the most accurate predictions across the validation sets. The cross-validated misclassification rate for the full tree  $T$  is denoted  $R^{\text{cv}}(T)$ ; misclassification for a pruned tree  $T_{-t}$  is denoted  $R^{\text{cv}}(T_{-t})$ .

In estimating  $R^{\text{cv}}(T)$ , we recognize that there is sampling variability: depending on how we choose training samples and validation samples (and depending on what data were observed),  $\widehat{R^{\text{cv}}(T)}$  will fluctuate. Breiman et al. propose the computation of a standard error estimate to describe the variability in cross-validated predictive accuracy estimates. For tree  $T$ , this standard error arises from the variability in predictive accuracy:

$$s^2(\widehat{R^{\text{cv}}(T)}) = \frac{1}{N} \sum_{j=1}^N \left[ (y_j - \hat{y}_j)^2 - \widehat{R^{\text{cv}}(T)} \right]^2, \quad (4)$$

where  $\hat{y}_j$  is the predicted value of  $y_j$ , obtained when  $y_j$  is a member of the validation set. Equation (4) can be used for both continuous and binary (0/1) response. The standard error may then be computed familiarly as:

$$\text{SE}(\widehat{R^{\text{cv}}(T)}) = \sqrt{s^2(\widehat{R^{\text{cv}}(T)})/N}. \quad (5)$$

Breiman et al. (pp. 307–308) note that this is only a heuristic standard error estimate, because it falsely assumes that the  $\hat{y}_j$  are independent. Nonetheless, it is the equation that is utilized by software implementing the Breiman et al. algorithm.

To ease the selection of a best tree, results can be summarized in a table similar to Table 2. This table contains results for the Boston housing data described above. The first column shows the number of splits in each tree, which is equivalent to the number of terminal nodes. The second column shows the error of each tree’s predictions on the training data (i.e., the data from which the tree was built) relative to the tree with 0 splits (i.e., the tree that makes the same prediction for every observation). The third column shows the error of each tree’s predictions on the validation data relative to the tree with 0 splits. The fourth column displays the estimated standard error of the validation-set misclassification rate (i.e., Equation (5)). Finally, the fifth and sixth columns show changes in number of splits and validation error, as compared to the tree with lowest validation error.

Breiman et al. (1984) suggest choosing the smallest tree whose validation error is within one standard error of the lowest validation error. This rule is intended to yield the smallest tree that still has good predictive accuracy. The rule is subjective (based on the experience of Breiman et al.), however, and others may argue for a different-sized tree. In our judgment, expert knowledge of the content area and notions of “practical significance” should play a large role here.

Using the Breiman et al. rule with Table 2, we would choose the smallest tree whose validation error is no larger than  $0.291 + 0.044 = 0.335$ . This leads us to choose the tree with 6 splits as the one with best generalizability, although, in this case, the tree with 5 splits is also close to the cutoff. The tree that we present in Figure 1 is the one with 6 splits.

### *Interpreting the Tree*

Once the tree with best generalizability is chosen, its interpretation is relatively straightforward. Consider Figure 1, which shows the Boston housing tree. We can immediately see that the average number of rooms per house (`Room`) has a large impact on price: the right branches of the tree all deal with neighborhoods that average 6.9 rooms or more, and the predicted prices in these neighborhoods are some of the largest in the tree. This implies that, regardless of anything else, large houses will be worth a large amount of money.

For smaller houses (those in neighborhoods that average fewer than 6.9 rooms), the left side of the tree shows that other variables are helpful for determining the house values. First, the percentage of low-class individuals in a neighborhood (`Low.class`) has an impact on housing prices: if that percentage is at least 14.4, then houses are predicted to be relatively cheap (\$119,784 for high-crime neighborhoods (`Crime`  $\geq$  7) and \$171,376 for low-crime neighborhoods (`Crime`  $<$  7)). If the percentage is below 14.4, then the distance from the house to the city (`Dist`) is important: houses close to the city are expensive (\$380,000), while the prices of houses further from the city again depend on number of rooms.

Now that we have described the basics behind BRP analysis, we will illustrate its use on an experimental dataset. In the experiment that we describe, researchers were interested in examining the effects of many demographic variables on an experimental response variable. The type of dataset that we use is common to many areas of psychology: researchers often collect many demographic variables from participants and present them as primary or secondary analyses in their publications.

### Application: Malpractice Verdicts

Diagnostic support systems (DSS) help individuals choose between multiple treatment options. They have been shown to decrease medical errors and outperform physicians on many diagnostic tasks, but they are grossly underutilized. One reason for this underutilization is the potential impact of DSS on medical malpractice verdicts. Consequently, the data that we describe here were collected to investigate the effect of DSS on medical malpractice verdicts. The data come from a study by Arkes, Shaffer, and Medow (2008).

#### *Data*

A DVD of a mock malpractice trial was created and shown to a US sample of 657 jury-eligible adults. The trial involved a patient (the plaintiff) who went to the emergency room complaining of abdominal pain. She was treated by a doctor (the defendant) who suspected the patient had appendicitis. Eight different versions of this mock malpractice trial were created, yielding three independent variables in a  $2 \times 2 \times 2$  between-subjects design. The three independent variables were: (1) whether or not the doctor used a DSS; (2) severity of the patient's symptoms; and (3) doctor's concordance with the DSS (i.e., whether or not the doctor followed the decision aid's recommendation). In the case where a DSS was not used, the third independent variable (concordance) is based on what the DSS would have recommended, if present.

Following presentation of the mock malpractice trial, the "jurors" were asked to indicate whether or not they thought the doctor was guilty of medical malpractice. Therefore, each participant entered a verdict of guilty or not guilty. In addition, participants responded to the following demographic questions: gender, age, education level, race/ethnicity, housing type, ownership status of living quarters, dual-income status, household income, head of household status, household size, employment status, presence

of children of various age groups, presence of adults, US region of residence, urban/rural residence, and presence of internet access.

Along with the three experimental independent variables, researchers were interested in analyzing effects of the demographic variables on jurors' verdicts (guilty/not guilty). This results in a total of 16 predictor variables. We describe a BRP analysis of jurors' verdicts using these 16 predictor variables below.

### *Verdicts*

As previously described, the first step in BRP involves building a large tree. For the verdict data, we allowed the tree to grow until 7 or fewer observations ended up in a terminal node. The resulting tree is displayed in Figure 3; it contains 32 terminal nodes. The 0's and 1's at the endpoints of the tree represent predicted "not guilty" and "guilty" verdicts, respectively. The verdict predictions arising from this tree are correct 77% of the time (compared to a base rate of 60% saying "not guilty") for the training dataset, but the tree overfits the data and will not be as accurate in predicting validation data. The tree needs to be pruned prior to interpretation.

The next step of BRP involves pruning the original tree and examining generalizability via cross-validation. From this step, we find that the large tree in Figure 3 does a poor job of predicting new data. While it was 77% accurate for the training data, it is only 59% accurate for validation (i.e., new) data. This is lower than the base rate of 60% not guilty verdicts, rendering the large tree useless. The smaller tree that we choose to interpret is displayed in Figure 4; it is 65% accurate for the training data and 63% accurate for the validation data. This is not a large improvement over the base rate, but the tree can still be informative of the predictor variables related to verdict.

Finally, our third step of BRP was interpretation. In Figure 4, the "guilty/not guilty" terms in each node represent the prediction for observations falling in that node.

The two numbers in each node report the observed number of “not guilty” voters and “guilty” voters, respectively, that fell in the node. The tree first splits the data based on ethnicity: the “caucasian” and “other” categories split to the left, while the “african-american,” “hispanic,” and “biracial” categories split to the right (these were the only five categories available to participants). The minority ethnicities (those split to the right) are split by age, with the younger people then being split based on the doctor’s concordance with the DSS. The only group of people who tended to vote “guilty” were minority ethnicities younger than retirement age. Even then, they only tended to vote “guilty” when the doctor defied the DSS. The tree provides a rich description of how the variables interact to influence a juror’s verdict.

### *Discussion*

For the mock juror verdicts, BRP yields interpretable results of high-dimensional data in a quick and straightforward fashion. Through the use of cross-validation, the analysis also gives us an idea of the generalizability of our results. In the original analyses, Arkes et al. (2008) fit many models with single predictor variables for variable selection, and they then fit a final logistic regression model to the selected predictors. Their final results generally agree with those presented here. Compared to the original analyses, BRP has a more automatic variable selection procedure, and the resulting tree may be more intuitive than logistic regression to some researchers. Finally, the fact that the BRP results are similar to the logistic regression results can give the researchers more confidence in their conclusions.

In the next section, we use simulated data to more generally compare BRP with regression.

### Contrasting BRP and Regression

Nonparametric methods such as BRP typically provide robust results at the potential sacrifice of precision. This is because BRP makes fewer assumptions than parametric methods, and it also treats continuous measures as ordinal (implying that BRP neglects information inherent in the data). If BRP used significance tests, we might show that BRP is less powerful than parametric methods in some situations, but more robust to assumption violations in other situations. BRP involves no significance tests, however, so there is no concept of “power” here. Instead, we can compare BRP’s predictive accuracy with regression’s predictive accuracy in different situations. In this section, we offer such a comparison via simulation, focusing on the regression assumption of linearity (i.e., linear relationship between predictor and response variable). The simulations also demonstrate the performance of BRP when there is only one predictor variable; traditional applications of BRP typically involve many predictor variables.

#### *Method*

We generated data from two simple models: one linear model and one nonlinear model. The linear model was  $y = x + e$ , with  $e \sim N(0, 1)$ . The nonlinear model was  $y = 5^x + e$ , with  $e \sim N(0, 25)$ . The specific model parameters were chosen to clearly demonstrate the trade-off between robustness and precision described in the previous paragraph.

To compare BRP and regression, we first generated 1,000 samples from each of the two models. This was done for sample sizes of 50, 100, 500, 1000, and 5000. We then fit a regression model and a BRP model to each generated dataset. Finally, we examined the fitted models’ abilities to predict a new dataset that matched the attributes (sample size and generating model) of the data to which the models were originally fit. This is essentially a cross-validation procedure with a single training set and a single validation

set. We measured the models' predictive abilities on the validation data via  $R^2$ , and we summarize the models' abilities below via median  $R^2$  across the 1,000 validation datasets for each sample size.

### *Results*

Figure 5 contains two graphs, the first of which contains results for data generated from the linear model and the second of which contains results for data generated from the nonlinear model. The x-axis reflects the sizes of the generated datasets, and the separate lines display median  $R^2$ 's of the fitted models (when predicting new data). The variability in the medians is very small (based on sample sizes of 1,000 each), so error bars are excluded from the graphs.

The first graph (data generated from the linear model) shows that the fitted regression model does an excellent job of predicting new regression data, regardless of sample size (regression's maximum attainable  $R^2$  in this example is 0.5). Regression predictions dominate BRP predictions at all sample sizes, demonstrating the "price" that BRP pays for making fewer assumptions. At large sample sizes ( $N \geq 500$ ), however, BRP's predictions are nearly as accurate as those of regression. This demonstrates that large sample sizes can offset any nonparametric "disadvantage" that BRP may possess. Further, if sample size is large enough, BRP can approximate a line.

The second graph (data generated from the nonlinear model) shows that BRP predictions dominate regression predictions at all sample sizes, and the difference between  $R^2$ 's increases with  $N$ . BRP's performance is better because it made no linearity assumptions. Conversely, the linearity violation led to poor regression predictions regardless of sample size. These two simple examples show that regression's extra assumptions can lead to improved or diminished predictions, depending on the attributes of the data. While there are specific applications where one method outperforms another

(e.g., Allore, Tinetti, Araujo, Hardy, & Peduzzi, 2005), it should be clear that neither method is optimal for all situations (Breiman et al., 1984).

The above results highlight the importance of checking model assumptions. In addition to existing methods for checking regression assumptions (e.g., robust regression, bootstrapping, nonparametric regression; see Neter, Kutner, Nachtsheim, & Wasserman, 1996 for an overview), it may be helpful to apply both regression and BRP to the same data. If BRP yields a considerably larger  $R^2$  than regression, then researchers may choose to rely on BRP as a primary analysis. Alternatively, researchers may use the BRP results to make informed modifications to a regression model (i.e., to transform variables or to select predictor variables).

### General Discussion

In this paper, we described and illustrated binary recursive partitioning methods for psychological data analysis. These methods place more emphasis on accurate prediction and less emphasis on a well-defined data model. BRP can generally be described as a search through all predictor variables to find the binary split that maximizes predictive accuracy. When that split is found, the procedure is repeated for the subgroups of data resulting from the original split. Repetition of this process yields a predictive tree that potentially incorporates many predictor variables. The tree can then be pruned, or reduced, in an attempt to maximize generalizability to future data. In our experience, the resulting tree is easy to display and understand.

BRP analyses can be conducted relatively automatically: the `rpart` package in R will carry out tree building, pruning, and cross-validation in a small number of commands (see Appendix B for an analysis of the Boston housing data). Other software that will carry out the analyses include CART, SPSS AnswerTree, and Systat.

### *BRP Difficulties*

The simulations in this paper showed that BRP's few assumptions place it at a disadvantage when regression assumptions (or other, more restrictive assumptions) are satisfied. There are also some technical difficulties associated with the procedure. For example, BRP has a selection bias towards predictor variables that support a greater number of splits (Shih, 2004). Consider the use of "gender" and "age" as predictor variables. There is only one possible split for "gender" (males go down one branch, females down the other), but there are many possible splits for "age" (assuming individuals of many ages are sampled). All else being equal, BRP would be more likely to select age than gender. Differing numbers of possible splits on predictor variables may also occur through ties or through missing data. To correct for this selection bias, it is possible to use a splitting criterion based on a chi-square statistic, with the best split taken to be the one with the smallest  $p$ -value (Loh, 2002; Hothorn, Hornik, & Zeileis, 2006). The  $p$ -value does not directly yield final conclusions, but it is used as an intermediate step to construct a less-biased tree. The resulting tree retains its intuitive nature, being similar to the others displayed in this paper.

Second, as Dusseldorp and Meulman (2004) discuss, BRP cannot immediately distinguish between main effects and interaction effects. Trees are made up of many splits, none of which are labeled "main effect" or "interaction." It is theoretically possible to distinguish between main effects and interactions in the following manner: starting with the initial split, consider the left branch separately from the right branch. If the same splits occur in both the left and right branch of the initial split, then the initial split is a main effect. If not, then there is an interaction effect. However, this sort of examination is only a heuristic and is unlikely to work for all predictor variables present in the tree.

To aid in distinguishing between main effects and interactions, it is possible to first fit a regression model that only includes main effects. BRP is then applied to the residuals

from the regression model, so that only interaction effects are left for BRP to estimate. Dusseldorp and Meulman's (2004) *regression trunk* procedure uses these ideas in the context of treatment-covariate interactions. The procedure first fits a main-effects regression model to the response variable, with the predictors assumed to be a binary treatment variable (i.e., treatment vs control groups) and other covariates. BRP is then applied to residuals from the regression model, with separate trees being fit to the treatment group and to the control group. Finally, the BRP results are used to create a new regression model with specific interaction terms. As shown in Appendix B, the regression trunk procedure is easily implemented in R or other programs that can perform both regression and BRP. Alternatively, Dusseldorp, Conversano, and Van Os (in press) present a similar procedure that simultaneously fits trees and regression models, with the trees accounting for interaction effects. This procedure, called STIMA, is implemented in its own R package.

A third difficulty of BRP is that the procedure searches for a local optimum instead of a global optimum. That is, at each step in the tree-building process, the procedure chooses the best split with no regard for future splits (i.e., it is a greedy search). There could be several good potential splits at any step in the procedure, but the tree will only choose the best one. One consequence of this is that a useful predictor variable can be completely left out of the resulting tree. Breiman et al. (1984) discuss the notion of "surrogate splits" and their use in determining the relative importance of predictor variables (even those that do not appear in the tree). Furthermore, there exist newer procedures (Chipman, George, & McCulloch, 1998) that use Bayesian methods to search for trees in a non-greedy fashion.<sup>7</sup>

### *BRP Extensions*

While space does not permit a thorough overview, there exist many modern statistical methods that are related to BRP. Many of these methods utilize BRP, while others accomplish similar tasks as BRP. The methods often yield improvements in predictive accuracy at the sacrifice of interpretability.

Methods that traditionally build off of BRP (though the ideas could be used with other models) include bagging, random forests, and boosting; social science introductions to these topics may be found in Berk (2006) and in Strobl, Malley, and Tutz (2009). Bagging (Breiman, 1996) fits trees to bootstrap samples of the original data. Overall predictions are averaged over the trees, often resulting in improved predictive accuracy. Random forests (Breiman, 2001a) are similar to bagging, except they utilize a random subset of predictor variables to create each split in each tree. This helps ensure that no single predictor variable dominates all others during tree construction, resulting in more diverse trees and potentially-better predictions. Boosting (Freund & Schapire, 1996; Friedman, 2001; see Friedman & Meulman, 2003 for a medical application) employs case weights to build a sequence of trees (or other statistical models), such that hard-to-predict observations are weighted heavily and easy-to-predict observations are weighted lightly. Overall predictions are obtained by taking a weighted average across individual tree predictions, where the weights here come from the predictive accuracy of each tree.

Other methods that may be of interest to the reader include model-based BRP (Zeileis, Hothorn, & Hornik, 2008), trees with probabilistic splits (Murthy, 1998; Yuan & Shaw, 1995), and, as mentioned previously, the STIMA procedure (Dusseldorp et al., in press). Finally, in a regression context, nonlinear optimal scaling methods (e.g., van der Kooij, Meulman, & Heiser, 2006) automatically transform predictor and response variables (both discrete and continuous) to optimize the regression fitting criterion (e.g., sum of squared error). They are not formally related to trees, though they maintain many

of tree advantages described in the introduction.

### *Summary*

In this article, we provided background on BRP and described its use in psychology. We also compared BRP to regression using some simple simulations. While BRP is not meant to replace regression, it is an alternative analysis that can be used to help verify regression results or to discover results that are undetectable using regression. More generally, BRP is one of many statistical methods that is made possible by fast computing. These methods are often very good for prediction, and they are becoming increasingly available in common statistical software. We encourage researchers to continue to consider ways in which these methods can improve psychological data analysis.

## References

Allore, H., Tinetti, M. E., Araujo, K. L. B., Hardy, S., & Peduzzi, P. (2005). A case study found that a regression tree outperformed multiple linear regression in predicting the relationship between impairments and social and productive activities scores. *Journal of Clinical Epidemiology*, 58, 154–161.

Arkes, H. R., Shaffer, V. A., & Medow, M. A. (2008). The influence of a physician's use of a diagnostic decision aid on the malpractice verdicts of mock jurors. *Medical Decision Making*, 28, 201–208.

Belsley, D. A., Kuh, E., & Welsch, R. E. (1980). *Regression diagnostics: Identifying influential data and sources of collinearity*. New York: Wiley.

Berk, R. A. (2006). An introduction to ensemble methods for data analysis. *Sociological Methods & Research*, 34, 263-295.

Bouroche, J. M., & Tennenhaus, M. (1971). Some segmentation methods. *Metra*, 7, 407–416.

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

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

Breiman, L. (2001b). Statistical modeling: The two cultures. *Statistical Science*, 16, 199-231.

Breiman, L., Friedman, J. H., Olshen, R. A., & Stone, C. J. (1984). *Classification and regression trees*. Belmont, CA: Wadsworth.

Browne, M. W. (2000). Cross-validation methods. *Journal of Mathematical Psychology*, 44, 108-132.

Chipman, H. A., George, E. I., & McCulloch, R. E. (1998). Bayesian CART model search. *Journal of the American Statistical Association*, 93, 935–948.

Dusseldorp, E., Conversano, C., & Van Os, B. J. (in press). Combining an additive and tree-based regression model simultaneously: STIMA. *Journal of Computational and Graphical Statistics*.

Dusseldorp, E., & Meulman, J. J. (2004). The regression trunk approach to discover treatment covariate interaction. *Psychometrika*, 69, 355-374.

Fielding, A. (1977). Binary segmentation: The automatic interaction detector and related techniques for exploring data structure. In C. A. O'Muircheartaigh & C. Payne (Eds.), *The analysis of survey data* (Vol. I, pp. 221-257). New York: Wiley.

Freund, Y., & Schapire, R. E. (1996). Experiments with a new boosting algorithm. In *Machine learning: Proceedings of the thirteenth international conference* (pp. 325-332). San Francisco: Morgan Kauffman.

Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, 29, 1189-1232.

Friedman, J. H., & Meulman, J. J. (2003). Multiple additive regression trees with application in epidemiology. *Statistics in Medicine*, 22, 1365-1381.

Harrison, D., & Rubenfeld, D. L. (1978). Hedonic prices and the demand for clean air. *Journal of Environmental Economics and Management*, 5, 81-102.

Hastie, T., Tibshirani, R., & Friedman, J. (2001). *The elements of statistical learning*. New York: Springer.

Hothorn, T., Hornik, K., & Zeileis, A. (2006). Unbiased recursive partitioning: A conditional inference framework. *Journal of Computational and Graphical Statistics*, 15, 651-674.

Loh, W.-Y. (2002). Regression trees with unbiased variable selection and interaction detection. *Statistica Sinica*, 12, 361-386.

Morgan, J. N., & Messenger, R. C. (1973). *THAID: A sequential analysis program for the analysis of nominal scale dependent variables*. Ann Arbor, MI: Institute for Social Research, University of Michigan.

Morgan, J. N., & Sonquist, J. A. (1963). Problems in the analysis of survey data, and a proposal. *Journal of the American Statistical Association*, 58, 415-434.

Murthy, S. K. (1998). Automatic construction of decision trees from data: A multi-disciplinary survey. *Data Mining and Knowledge Discovery*, 2, 345-389.

Neter, J., Kutner, M. H., Nachtsheim, C. J., & Wasserman, W. (1996). *Applied linear statistical models* (4th ed.). Boston: McGraw Hill.

R Development Core Team. (2008). *R: A language and environment for statistical computing*. Vienna, Austria. (ISBN 3-900051-07-0)

Ripley, B. D. (1996). *Pattern recognition and neural networks*. New York: Cambridge.

Shih, Y.-S. (2004). A note on split selection bias in classification trees. *Computational Statistics & Data Analysis*, 45, 457-466.

Sonquist, J. A. (1970). *Multivariate model building: The validation of a search strategy*. Ann Arbor, MI: Institute for Social Research, University of Michigan.

Sonquist, J. A., Baker, E. L., & Morgan, J. N. (1974). *Searching for structure* (Revised ed.). Ann Arbor, MI: Institute for Social Research, University of Michigan.

Strobl, C., Malley, J., & Tutz, G. (2009). An introduction to recursive partitioning: Rationale, application, and characteristics nad classification and regression trees, bagging, and random forests. *Psychological Methods*, 14, 323-348.

Therneau, T. M., Atkinson, B., & Ripley, B. (2006). *rpart: Recursive partitioning*. (R package version 3.1-29; R port by Brian Ripley)

van der Kooij, A. J., Meulman, J. J., & Heiser, W. J. (2006). Local minima in categorical multiple regression. *Computational Statistics & Data Analysis*, 50, 446–462.

Venables, W. N., & Ripley, B. D. (2002). *Modern applied statistics with S* (Fourth ed.). New York: Springer.

Yuan, Y., & Shaw, M. J. (1995). Induction of fuzzy decision trees. *Fuzzy Sets and Systems*, 69, 125-139.

Zeileis, A., Hothorn, T., & Hornik, K. (2008). Model-based recursive partitioning. *Journal of Computational and Graphical Statistics*, 17, 492–514.

Zhang, H., & Singer, B. (1999). *Recursive partitioning in the health sciences*. New York: Springer.

## Appendix A

### BRP with Continuous Response Variables

This appendix contains details for BRP with continuous response variables; the focus of the text was on BRP with categorical response variables.

For BRP with continuous response, we must define a new impurity measure for selecting splits (see Equation (1)). A natural choice for an impurity measure is one based on the sum of squared error within a node:

$$i(t) = \sum_{i=1}^{n_t} (y_i - \bar{y}_t)^2, \quad (6)$$

where  $n_t$  is the number of observations in node  $t$  and  $\bar{y}_t$  is the mean of the observations in node  $t$ .

Determination of an optimal split proceeds in a similar manner to categorical  $y$ 's. Starting with a node  $t$ , we seek a new split  $s$  that forms two new nodes,  $t_L$  and  $t_R$ . The  $s$  that we choose will be the one that leads to the greatest decrease in within-node sums of squares:

$$\Delta i(s, t) = i(t) - i(t_L) - i(t_R). \quad (7)$$

The prediction for observations in node  $t$  is then  $\bar{y}_t$ .

It is important to note that the sum of squared error is also used to prune the tree (see Equation (3);  $R(T)$  would now be the sum of squared error for tree  $T$ ). This is different from BRP with categorical response variables, where one criterion (e.g., Gini) is used to build the tree and misclassification rate is used to prune the tree.

## Appendix B

### R Code for BRP Analysis and Regression Trunks

The following commands can be used in R to replicate our example with the Boston housing data. These commands assume that the add-on packages of `MASS` and `rpart` have been installed; if they have not been installed, the command `install.packages()` can be used. For more information on this command, type `?install.packages()`.

```
library(MASS) # Load MASS package
library(rpart) # Load rpart package

# Build trees of different sizes
bost.tree <- rpart(medv ~ ., data=Boston)

# Select the "best" tree; this command will display
# a table that is similar to Table 1 of this manuscript.
# The xerror column is "Validation error";
# the xstd column is "SE(Validation error)"
bost.tree$cptable

# Based on the table above, obtain the "best" tree.
# We chose the tree in the seventh row of the table; that is
# where the "7" in the command below comes from.
bost.prune <- prune(bost.tree, cp=bost.tree$cptable[7,1])

# Visualize the tree. For more plotting options, see ?plot.rpart() and
# ?text.rpart().
plot(bost.prune); text(bost.prune)
```

To carry out Dusseldorp and Meulman's (2004) regression trunk procedure, the residuals from a regression are sent to `rpart`. Continuing with the Boston housing data, the following example uses a regression of housing value on crime rate, distance from neighborhood to major employment, and whether or not the neighborhood is on the Charles River. The Charles River variable is analogous to a treatment variable in Dusseldorp and Meulman's framework.

```
# Fit regression model:
```

```
reg.mod <- lm(medv ~ crim+dis+chas, data=Boston)

# Obtain residuals:
medv.resid <- reg.mod$residuals

# Apply BRP to residuals separately for neighborhoods
# on the river and neighborhoods off the river.
brp.1 <- rpart(medv.resid ~ crim+dis, data=Boston, subset=chas==1)
brp.0 <- rpart(medv.resid ~ crim+dis, data=Boston, subset=chas==0)

# Inspect trees
brp.1$cptable
brp.0$cptable
```

For this dataset, the cross-validation results imply no interactions between the Charles River variable and either of the covariates. If there were interactions, dummy variables reflecting the trees could then be entered into a revised regression model.

**Author Note**

This work benefited from the **rpart** package (Therneau, Atkinson, & Ripley, 2006) and **MASS** package (Venables & Ripley, 2002) available for **R** (R Development Core Team, 2008). The authors thank Hal R. Arkes and Mitchell A. Medow for kindly lending their data for use in this paper. The authors also thank Jerome Friedman, Jacqueline Meulman, an anonymous reviewer, and the editor for comments that improved the quality of the paper. Correspondence concerning this article can be sent to Edgar C. Merkle, Department of Psychology, 1845 Fairmount Box 34, Wichita, KS 67260, [edgar.merkle@wichita.edu](mailto:edgar.merkle@wichita.edu).

## Footnotes

<sup>1</sup>The CART acronym is trademarked by Salford Systems, so, to prevent “trademark genericide,” we use the term “binary recursive partitioning.”

<sup>2</sup>We wrote our BRP program using R. While the BRP code in existing R packages (`tree`, `rpart`) is more efficient, our code may be easier to follow because: (1) our code does not link to C; and (2) we have tried to add many comments. This code can be obtained by contacting the first author.

<sup>3</sup>Breiman et al. (1984) present a similar analysis using the same dataset. We replicate the analysis here because the Boston housing data are freely available in R’s MASS package. Thus, readers have the ability to replicate the analysis using the code in our Appendix B.

<sup>4</sup>The actual number of splits examined is usually smaller, depending on the specific “stopping rules” that are employed. For example, we may only wish to consider splits that result in more than 5 observations (say) in  $t_L$  and  $t_R$ . If no such splits exist, then the tree-building procedure stops. Some further details on stopping rules appear below.

<sup>5</sup>These options exist in the `rpart` package as `minsplit` and `cp`, respectively.

<sup>6</sup>The algorithm actually matches trees to one another based on values of  $G(t)$  that are obtained in the pruning process. We refer the interested reader to pp. 75–78 of Breiman et al. (1984) for more detail.

<sup>7</sup>While these methods are not greedy searches, they do not consider the entire space of possible trees, either. That remains computationally infeasible.

Table 1  
*Pruning calculations for a hypothetical tree.*

Tree	Misclassification rate	Terminal nodes	$G(t)$
T	.6	10	—
1	.63	9	.030
2	.75	7	.050
3	.83	5	.046
4	.96	2	.045

Table 2  
*Cross-validation results from pruning the Boston housing tree.*

Splits	Training error	Validation error	SE(Validation error)	$\Delta$ (splits)	$\Delta$ (Validation error)
0	1.000	1.000	0.083	7	0.709
1	0.547	0.654	0.060	6	0.363
2	0.376	0.443	0.049	5	0.152
3	0.304	0.356	0.044	4	0.065
4	0.268	0.346	0.045	3	0.055
5	0.235	0.338	0.046	2	0.047
6	0.208	0.292*	0.042	1	0.001
7	0.192	0.291	0.044		

*Note: Following the one-standard error rule, the smallest tree with validation error below  $.291 + .044 = .335$  is chosen.*

*This leads us to choose the starred value (thus, the tree with 6 splits).*

### Figure Captions

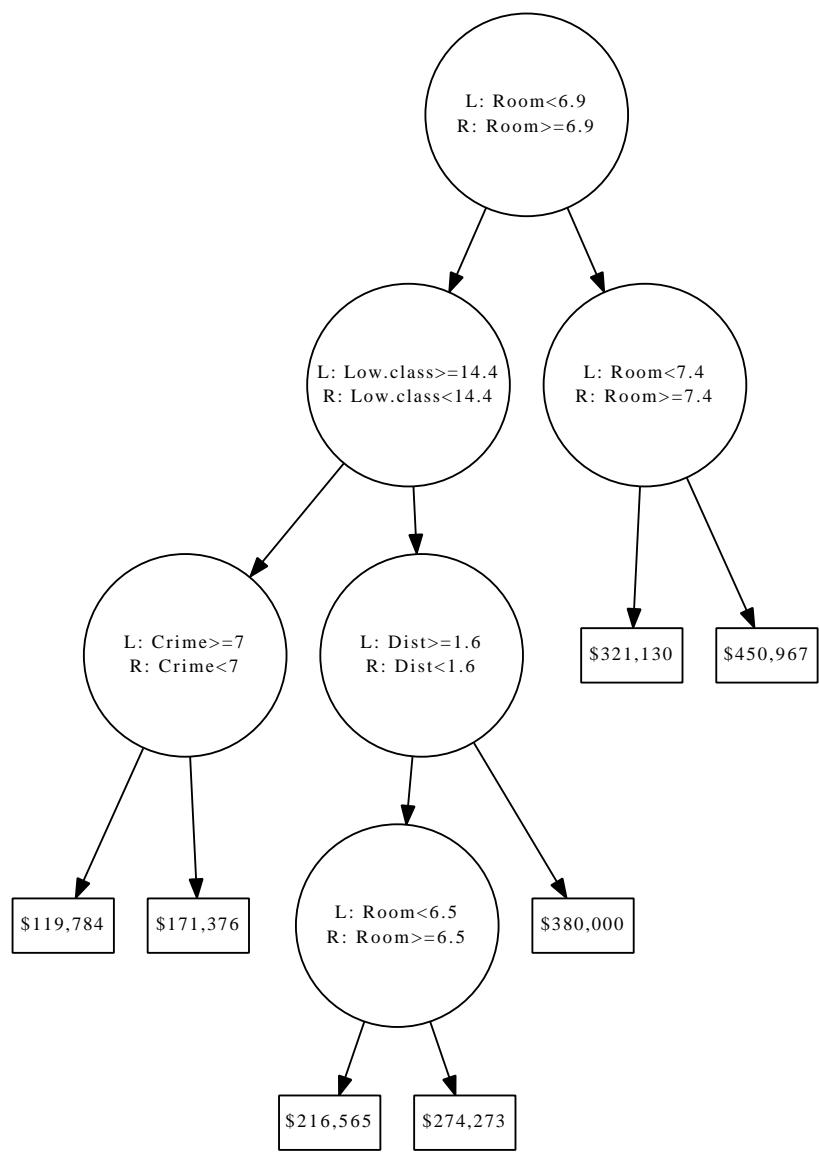
*Figure 1.* Tree for the Boston housing data. Observations follow the left (L) or right (R) branches of tree based on the conditions in each node. Predictions of a neighborhood's median home price is made once an observation reaches a terminal node.

*Figure 2.* The Gini index for response variables with two classes. When the proportion of observations in class 1 is .5, the index is at its maximum. When the proportion of observations in class 1 is 0 or 1 (reflecting a "pure" node), the index is at its minimum of 0.

*Figure 3.* Unpruned tree for mock jurors' verdicts. The tree is displayed for demonstration purposes; it is not to be interpreted prior to pruning.

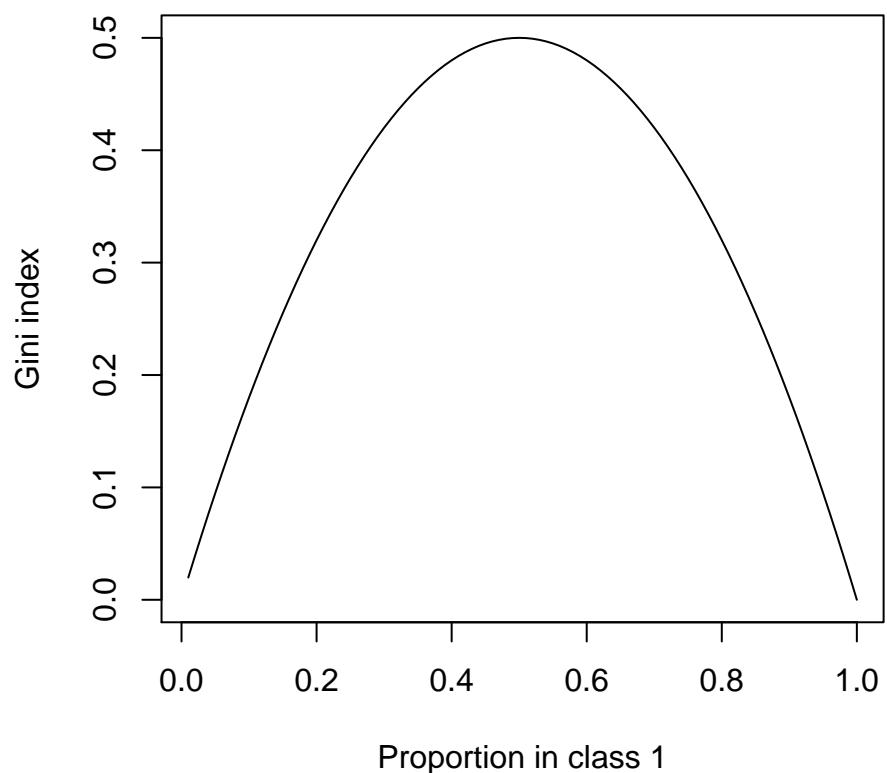
*Figure 4.* Tree for mock jurors' verdicts.

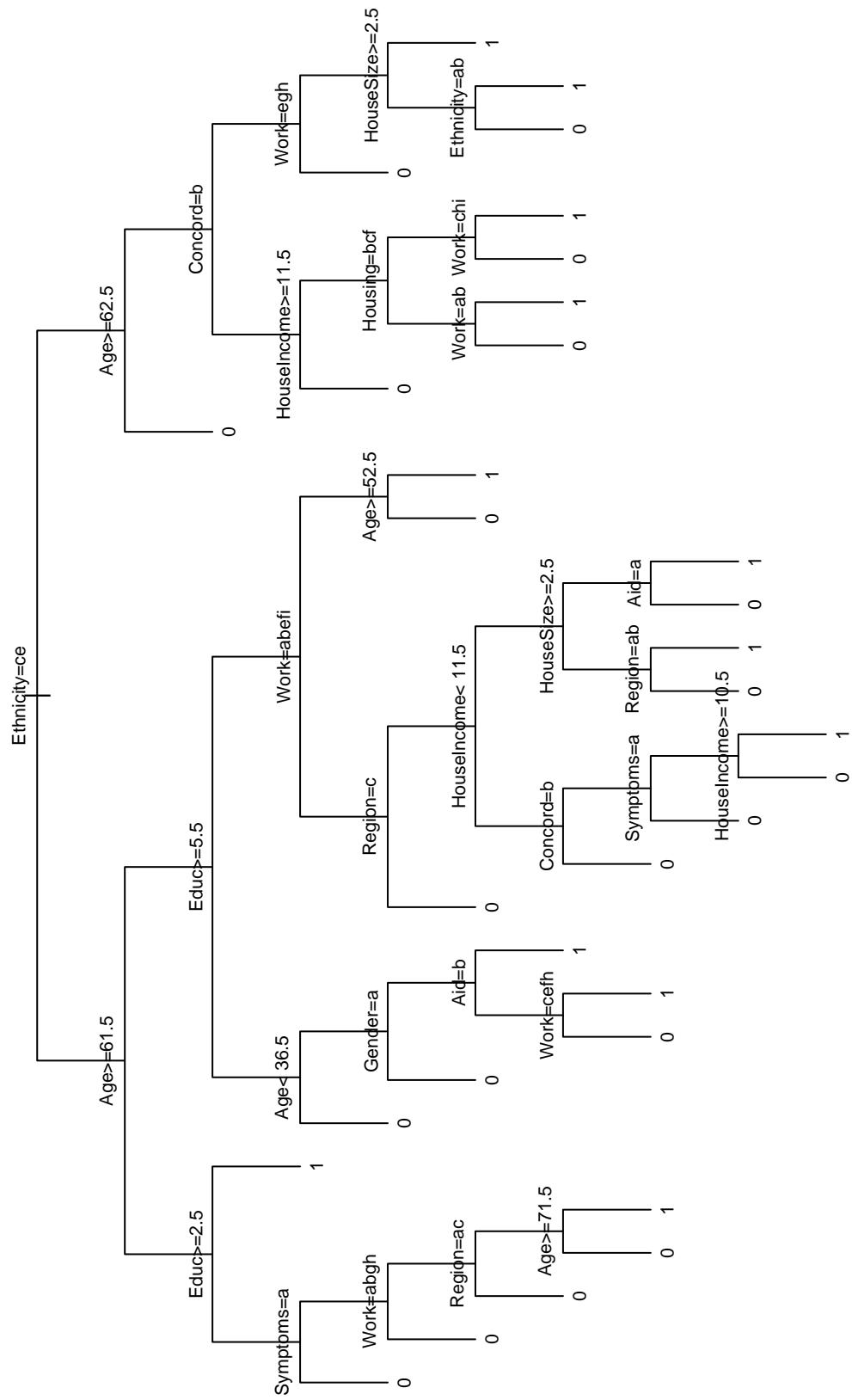
*Figure 5.* Median cross-validated  $R^2$ 's for regression and BRP models at various sample sizes. The top figure contains results for data generated from a linear model, and the bottom figure contains results for data generated from a nonlinear model.

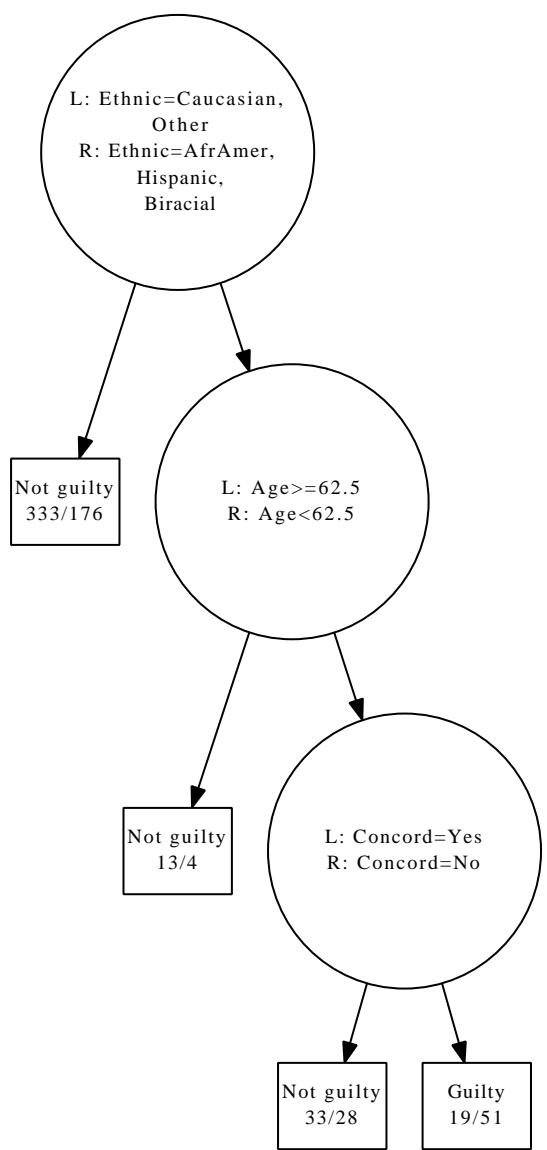


Note: Variable abbreviations, measured for each neighborhood, are: *Room*=average number of rooms per house; *Low.class*=percentage of lower-class individuals; *Crime*=crime rate; *Dist*=distance to major employment centers.

**Figure 2**

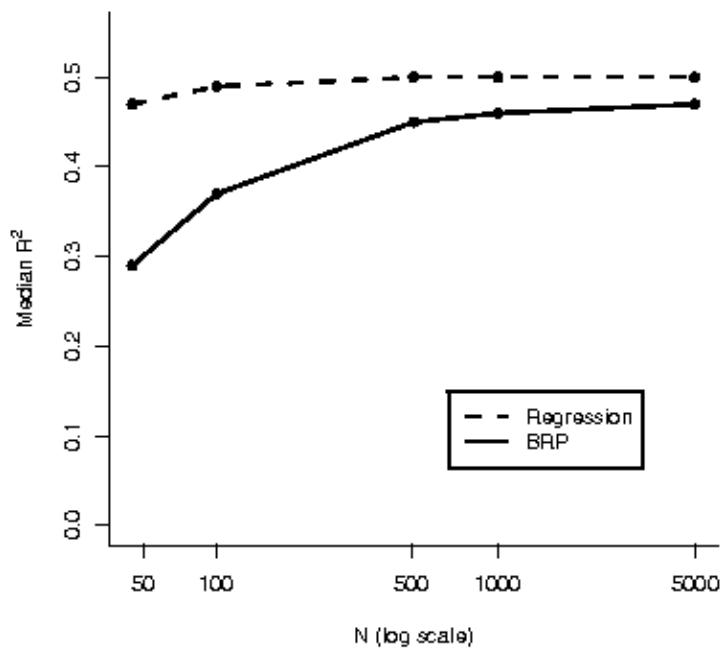






*Note: Concord equals Yes if the doctor followed the recommendation of the decision aid and No if the doctor did not follow the recommendation.*

Linear Data



Nonlinear Data

