FIRM: Formal Inference-based Recursive Modeling

Overview

Recursive modeling is an attractive data-analytic tool for studying the relationship between a dependent variable and a collection of predictor variables. In this, it complements methods such as multiple regression, analysis of variance, neural nets, generalized additive models, discriminant analysis and log linear modeling. It is particularly helpful when simple models like regression do not work, as happens when the relationship connecting the variables is not straightforward—for example if there are synergies between the predictors. It is attractive when there is missing information in the data. It can also be valuable when used in conjunction with a more traditional statistical data analysis, as it may provide evidence that the model assumptions underlying the traditional analysis are valid, something that is otherwise not easy to do.

The output of a recursive model analysis is a "dendrogram", or "tree diagram". This shows how the predictors can be used to partition the data into successively smaller and more homogeneous subgroups. This dendrogram can be used as an immediate pictorial way of making predictions for future cases, or "interpreted" in terms of the effects on the dependent variable of changes in the predictor. The maximum problem size the codes distributed in this package can handle is 1000 predictors; the number of cases is limited only by the available disk space.

Recursive partitioning (RP) is not attractive for all data sets. Going down a tree leads to rapidly diminishing sample sizes and the analysis comes to an end quite quickly if the initial sample size was small. As a rough guide, the sample size needs to be in three digits before recursive partitioning is likely to be worth trying. Like all feature selection methods, it is also highly non-robust in the sense that small changes in the data can lead to large changes in the output. The results of a RP analysis therefore always need to be thought of as *a* model for the data, rather than as *the* model for the data. Within these limitations though, the recursive partitioning analysis gives an easy, quick way of getting a picture of the relationship that may be quite different than that given by traditional methods, and that is very informative when it is different.

FIRM is made available through LISREL by the kind permission of the author, Professor Douglas M. Hawkins, Department of Applied Statistics, University of Minnesota.

Recursive Partitioning Concepts

The most common methods of investigating the relationship between a dependent variable Y and a set of predictors $X_1, X_2, \ldots, X_p$ are the linear model (multiple regression, analysis of variance and analysis of covariance) for a continuous dependent variable, and the log linear model for a categorical dependent. Both methods involve strong model assumptions. For example, multiple regression assumes that the relationship between Y and the X_i is linear, that there are no interaction terms other than those explicitly included in the model; and that the residuals satisfy strong statistical assumptions of a normal distribution and constant variance. The log linear model for categorical data assumes that any interaction terms not explicitly included are not needed; while it provides an overall χ^2 test-of-fit of the model, when this test fails it does not help to identify what interaction terms are needed. Discriminant analysis is widely used to find rules to classify observations into the different populations from which they came. It is another example of a methodology with strong model assumptions that are not easy to check.

A different and in some ways diametrically opposite approach to all these problems is modeling by recursive partitioning. In this approach, the calibration data set is successively split into ever-smaller subsets based on the values of the predictor variables. Each split is designed to separate the cases in the node being split into a set of successor groups, which are in some sense maximally internally homogeneous.

An example of a data set in which FIRM is a potential method of analysis is the "head injuries" data set of Titterington *et al* (1981) (data kindly supplied by Professor Titterington). As we will be using this data set to illustrate the operation of the FIRM codes, and since the data set is included in the FIRM distribution for testing purposes, it may be appropriate to say something about it. The data set was gathered in an attempt to predict the final outcome of 500 hospital patients who had suffered head injuries. The outcome for each patient was that he or she was:

- *Dead or vegetative* (52% of the cases);
- had *severe* disabilities (10% of the cases);
- or had a *moderate or good* recovery (38% of the cases).

This outcome is to be predicted on the basis of 6 predictors assessed on the patients' admission to the hospital:

- *Age* : The age of the patient. This was grouped into decades in the original data, and is grouped the same way here. It has eight classes.
- *EMV* : This is a composite score of three measures—of eye-opening in response to stimulation; motor response of best limb; and verbal response. This has seven classes, but is not measured in all cases, so that there are eight possible codes for this score, these being the seven measurements and an eighth "missing" category.
- *MRP* : This is a composite score of motor responses in all four limbs. This also has seven measurement classes with an eighth class for missing information.
- *Change* : The change in neurological function over the first 24 hours. This was graded 1, 2 or 3, with a fourth class for missing information.
- *Eye indicator* : A summary of diagnostics on the eyes. This too had three measurement classes, with a fourth for missing information.
- *Pupils* : Eye pupil reaction to light, namely present, absent, or missing.

CATFIRM

Figure 1 is a dendrogram showing the end result of analyzing the data set using the CATFIRM syntax, which is used for a categorical dependent variable like this one. Syntax can be found in the file **headicat.pr2**.
The dendrogram produced by CATFIRM is a very informative picture of the relationship between these predictors and the patients' final outcome. We see in it that the most significant separation was obtained by splitting the full sample on the basis of the predictor *Pupils*.

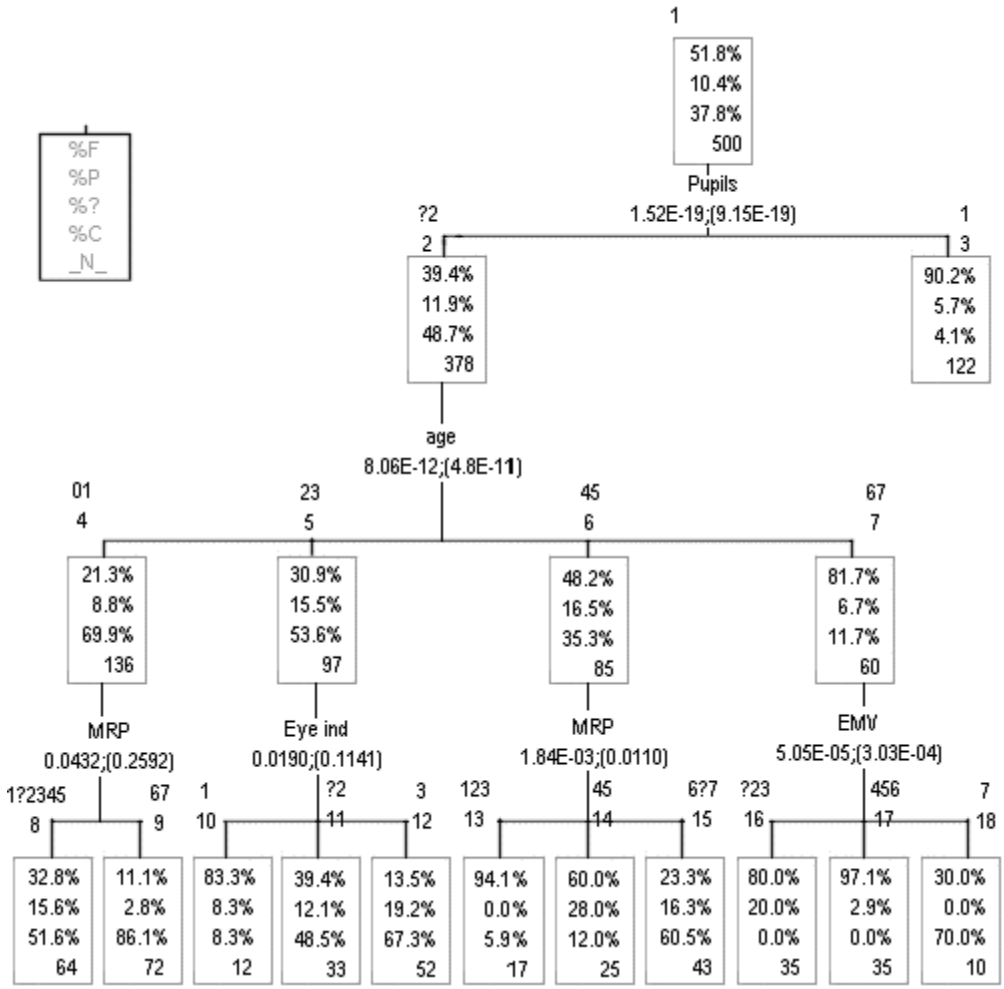

Figure 1: CATFIRM dendrogram of "head injuries" data set

The groups of 378 cases for which *Pupils* had the value 2 or the value ? (that is, missing) were statistically indistinguishable, and so are grouped together. They constitute one of the successor groups (node number 2), while those 122 for whom *Pupils* had the value 1 constitute the other (node number 3), a group with much worse outcomes—90% dead or vegetative compared with 39% of those in node 2.

Each of these nodes in turn is subjected to the same analysis. The cases in node number 2 can be split again into more homogeneous subgroups. The most significant such split is obtained by separating the cases into four groups on the basis of the predictor *Age*. These are patients under 20 years old, (node 4), patients 20 to 40 years old (node 5), those 40 to 60 years old (node 6) and those over 60 (node 7). The prognosis of these patients deteriorates with increasing age; 70% of those under 20 ended with moderate or good recoveries, while only 12% of those over 60 did.

Node 3 is terminal. Its cases cannot (at the significance levels selected) be split further. Node 4 can be split using *MRP*. Cases with *MRP* = 6 or 7 constitute a group with a favorable prognosis (86% with moderate to good recovery); and the other MRP levels constitute a less-favorable group but still better than average.

These groups, and their descendants, are analyzed in turn in the same way. Ultimately no further splits can be made and the analysis stops. Altogether 18 nodes are formed, of which 12 are terminal and the remaining 6 intermediate. Each split in the dendrogram shows the variable used to make the split and the values of the splitting variable that define each descendant node. It also lists the statistical significance (*p*-value) of the split. Two *p*-values are given: that on the left is FIRM's conservative *p*-value for the split. The *p*-value on the right is a Bonferroni-corrected value, reflecting the fact that the split actually made had the smallest *p*-value of all the predictors available to split that node. So for example, on the first split, the actual split made has a *p*-value of 1.52×10^{-19}. But when we recognize that this was selected because it was the most significant of the 6 possible splits, we may want to scale up its *p*-value by the factor 6 to allow for the fact that it was the smallest of 6 *p*-values available (one for each predictor). This gives its conservative Bonferroni-adjusted *p*-value as 9.15×10^{-19}.

The dendrogram and the analysis giving rise to it can be used in two obvious ways: for making predictions, and to gain understanding of the importance of and interrelationships between the different predictors. Taking the prediction use first, the dendrogram provides a quick and convenient way of predicting the outcome for a patient. Following patients down the tree to see into which terminal node they fall yields 12 typical patient profiles ranging from 96% *dead/vegetative* to 100% with *moderate to good* recoveries. The dendrogram is often used in exactly this way to make predictions for individual cases. Unlike, say, predictions made using multiple regression or discriminant analysis, the prediction requires no arithmetic—merely the ability to take a future case "down the tree". This allows for quick predictions with limited potential for arithmetic errors. A hospital could, for example, use Figure 1 to give an immediate estimate of a head-injured patient's prognosis.

The second use of the FIRM analysis is to gain some understanding of the predictive power of the different variables used and how they interrelate. There are many aspects to this analysis. An obvious one is to look at the dendrogram, seeing which predictors are used at the different levels. A rather deeper analysis is to look also at the predictors that were not used to make the split in each node, and see whether they could have discriminated between different subsets of cases. This analysis is done using the "summary" file part of the output, which can be requested by adding the keyword `sum` on the first line of the syntax file. In this head injuries data set, the summary file shows that all 6 predictors are very discriminating in the full data set, but are much less so as soon as the initial split has been performed. This means that there is a lot of overlap in their

predictive information: that different predictors are tapping into the same or related basic physiological indicators of the patient's state.

Another feature often seen is an *interaction* in which one predictor is predictive in one node, and a different one is predictive in another. This may be seen in this data set, where *Age* is used to split Node 4, but *Eye ind* is used to split Node 5. Using different predictors at different nodes at the same level of the dendrogram is one example of the interactions that motivated the ancestor of current recursive partitioning codes—the Automatic Interaction Detector (AID). Examples of diagnosing interactions from the dendrogram are given in Hawkins and Kass (1982).

CONFIRM

Figure 1 illustrates the use of CATFIRM, which is appropriate for a categorical dependent variable. The other analysis covered by the FIRM package is CONFIRM, which is used to study a dependent variable on the interval scale of measurement. Figure 2 shows an analysis of the same data using CONFIRM. The dependent variable was on a three-point ordinal scale, and for the CONFIRM analysis we treated it as being on the interval scale of measurement with values 1, 2 and 3 scoring 1 for the outcome "*dead or vegetative*"; 2 for the outcome "*severe disability*" and 3 for the outcome "*moderate to good recovery*". As the outcome is on the ordinal scale, this equal-spaced scale is not necessarily statistically appropriate in this data set and is used as a matter of convenience rather than with the implication that it may be the best way to proceed.

The full data set of 500 cases has a mean outcome score of 1.8600. The first split is made on the basis of the predictor *Pupils*. Cases for which *Pupils* is 1 or ? constitute the first descendant group (Node 2), while those for which it is 2 give Node 3. Note that this is the same predictor that CATFIRM elected to use, but the patients with missing values of *Pupils* are grouped with *Pupils* = 1 by the CONFIRM analysis, and with *Pupils* = 2 by CATFIRM. Going to the descendant groups, Node 2 is then split on *MRP* (unlike in CATFIRM, where the corresponding node was terminal), while Node 3 is again split four ways on *Age*. The overall tree is bigger than that produced by CATFIRM—13 terminal and 8 interior nodes—and produces groups with prognostic scores ranging from a grim 1.0734 up to 2.8095 on the 1 to 3 scale. As with CATFIRM, the means in these terminal nodes could be used for prediction of future cases, giving estimates of the score that patients in the terminal nodes would have.

The statistical model used in CONFIRM can be described as a *piecewise constant regression* model. For example, among the patients in Node 3, the outcome is predicted to be 2.48 for patients aged under 20, 2.25 for those aged 20 to 40, 1.88 for those aged 40 to 60, and 1.32 for those aged over 60. This approach contrasts sharply with, say, a linear regression in which the outcome would be predicted to rise by some constant amount

with each additional year of *Age*. While this piecewise constant regression model is seldom valid exactly, it is often a reasonable working approximation of reality.

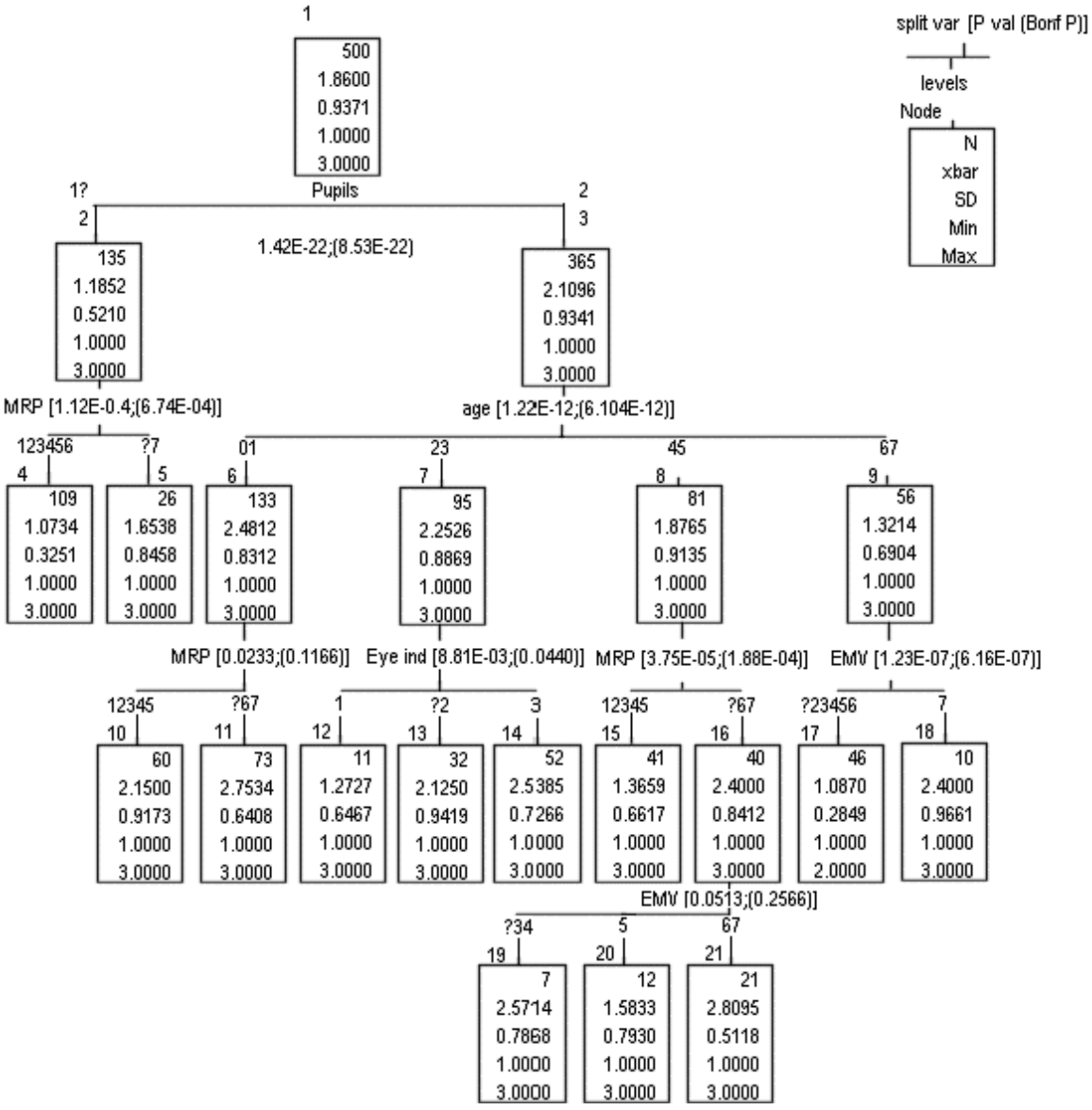

Figure 2: CONFIRM dendrogram of "head injuries" data set

Other Detailed Output Files

The dendrograms of Figures 1 and 2 produced by the FIRM codes are the most obviously and immediately useful output of a FIRM analysis. They are an extract of the much more detailed output, a sample of which is given in Chapter 6. This output contains the following (often very valuable) additional information:

- An analysis of each predictor in each node, showing which categories of the predictor FIRM finds it best to group together, and what the conservative statistical significance level of the split by each predictor is;
- The number of cases owing into each of the descendant groups;
- Summary statistics of the cases in the descendant nodes. In the case of CATFIRM, the summary statistics are a frequency breakdown of the cases between the different classes of the dependent variable. With CONFIRM, the summary statistics given are the arithmetic mean and standard deviation of the cases in the node. This summary information is echoed in the dendrogram.

FIRM Building Blocks

There are three elements to analysis by a recursive partitioning method such as FIRM:

- deciding which predictor to use to define the split;
- deciding which categories of the predictor should be grouped together so that the data set is not split more ways than are really necessary (for example, is a binary, three-way, four-way etc. split on this variable required); and
- deciding when to stop growing the tree.

Different implementations of recursive partitioning handle these issues in different ways. The theoretical basis for the procedures implemented in FIRM and used to produce these dendrograms is set out in detail in Hawkins and Kass (1982) and Kass (1980). Readers interested in the details of the methodology should refer to these two sources and to the papers they reference. The Ph.D. thesis by Kass (1975) included a PL/1 program for CHAID ("chi-squared automatic interaction detector")—CATFIRM is based on a translation of this code into FORTRAN. The methodology (but no code) is set out in a perhaps more widely accessible medium in the paper by Kass (1980). The 1981 technical report by Heymann discusses a code XAID ("extended automatic interaction detector") which extended the framework underlying CHAID to an interval-scale dependent variable. CONFIRM is a development of this program. The FIRM methodology has been applied to a wide variety of problems—for example, Hooton *et al* (1981) was an early application in medicine, and Hawkins, Young and Rusinko (1997) illustrates an application to drug discovery.

After this brief illustration of two FIRM runs, we will now go into some of the underlying ideas of the methodology. Readers who have not come across recursive partitioning before would be well advised to experiment with the CONFIRM and CATFIRM before going too far into this detail. See Chapter 6 for FIRM examples and also the **firmex** folder.

Predictor and Dependent Variable Types

Writings on measurement types often distinguish four scales of measurement:

- *Nominal*, in which the measurement divides the individuals studied into different classes. Eye color is an example of a nominal measure on a person.
- *Ordinal*, in which the different categories have some natural ordering. Social class is an example of an ordinal measure. So is a rating by the adjectives "Excellent", "Very good", "Good", "Fair", "Poor".
- *Interval*, in which a given difference between two values has the same meaning wherever these values are in the scale. Temperature is an example of a measurement on the interval scale—the temperature difference between 10 and 15 degrees is the same as that between 25 and 30 degrees.
- *Ratio*, which goes beyond the interval scale in having a natural zero point.

Each of these measurement scales adds something to the one above it, so an interval measure is also ordinal, but an ordinal measure is generally not interval.

FIRM can handle two types of dependent variable: those on the nominal scale (analyzed by CATFIRM), and those on the interval scale (analyzed by CONFIRM). There is no explicit provision for a dependent variable on the ordinal scale such as, for example, a qualitative rating "Excellent", "Good", "Fair" or "Poor". Dependents like this have to be handled either by ignoring the ordering information and treating them using CATFIRM, or by trying to find an appropriate score for the different categories. These scores could range from the naive (for example the 1-2-3 scoring we used to analyze the head injuries data with CONFIRM) to conceptually sounder scorings obtained by scaling methods.

Both FIRM approaches use the same predictor types and share a common underlying philosophy. While FIRM recognizes five predictor types, two going back to its Automatic Interaction Detection roots are fundamental:

- Nominal predictors (which are called "*free*" predictors in the AID terminology),
- Ordinal predictors (which AID labels "*monotonic*" predictors).

In practical use, some ordinal predictors are just categorical predictors with ordering among their scales, while others are interval-scaled. Suppose for example that you are studying automobile repair costs, and think the following predictors may be relevant:

- *make*: The make of the car. Let's suppose that in the study it is sensible to break this down into 5 groupings: GM, Ford, Chrysler, Japanese and European.
- *cyl*: The number of cylinders in the engine: 4, 6, 8 or 12.
- *price*: The manufacturer's suggested retail purchase price when new.
- *year*: The year of manufacture.

make is clearly nominal, and so would be used as a free predictor. In a linear model analysis, you would probably use analysis of variance, with *make* a factor. Looking at these predictors, *cyl* seems to be on the interval scale, and so might be used as an ordered (monotonic) predictor. If we were to use *cyl* in a linear regression study, we would be making a strong assumption that the repair costs difference between a 4- and an 8-cylinder car is the same as that between an 8- and a 12-cylinder car. FIRM's monotonic predictor type involves no such global assumption.

price is a ratio-scale predictor that would be specified as monotonic—it seems generally accepted that cars that are more expensive to buy are also more expensive to maintain. In fact, though, even if this were false, so long as the relationship between price and repair costs was reasonably smooth (and not necessarily even monotonic) FIRM's piecewise constant model fitting would be able to handle it.

Finally, *year* is an example of a predictor that looks to be on an interval scale (since a one-year gap is a one-year gap whether it is the gap between 1968 and 1969 or that between 1998 and 1999), but appearances may be deceptive. While some years' cars do certainly seem to be more expensive to maintain than other years' cars, it is not obvious that this relationship is a smooth one. A "lemon" year can be preceded and followed by years in which a manufacturer makes good, reliable cars. For this reason it would be preferable to make *year* a free predictor rather than monotonic. This will allow the cost to have any type of response to *year*, including one in which certain isolated years are bad while their neighbors are good. If the repair cost should turn out to vary smoothly with *year* then little would have been lost and that little could be recovered by repeating the run, respecifying *year* as monotonic.

Internal Grouping to get Starting Groups

Internally, the FIRM code requires all predictors to be reduced to a modest number (no more than 16 or 20) of distinct categories, and from this starting set it will then merge

categories until it gets to the final grouping. Thus in the car repair cost example, *make*, *cyl* and possibly *year* are ready to run. The predictor *price* is continuous—it likely takes on many different values. For FIRM analysis it would therefore have to be categorized into a set of price ranges. There are two ways of doing this: you may decide ahead of time what sensible price ranges are and use some other software to replace the dollar value of the price by a category number. This option is illustrated by the predictor *Age* in the "head injuries" data. A person's age is continuous, but in the original data set it had already been coded into decades before we got the data, and this coded value was used in the FIRM.

You can also have FIRM do the grouping for you. It will attempt to find nine initial working cutpoints, breaking the data range into 10 initial classes. The downside of this is that you may not like its selection of cutpoints: if we had had the subjects' actual ages in the head injuries data set, FIRM might have chosen strange-looking cutpoints like 14, 21, 27, 33, 43, 51, 59, 62, 68 and 81. These are not as neat or explainable as the ages by decades used by Titterington, *et al.*

Either way of breaking a continuous variable down into classes introduces some approximation. Breaking age into decades implies that only multiples of 10 years can be used as split points in the FIRM analysis. This leaves the possibility that, when age was used to split node 3 in CATFIRM with cut points at ages 20, 40 and 60, better fits might have been obtained had ages between the decade anniversaries been allowed as possible cut points: perhaps it would have been better to split at say age 18 rather than 20, 42 rather than 40 and 61 rather than 60. Maybe a completely different set of cut points could have been chosen. All this, whether likely or not, is certainly possible, and should be borne in mind when using continuous predictors in FIRM analysis.

Internally, all predictors in a FIRM analysis will have values that are consecutive integers, either because they started out that way or because you allowed FIRM to translate them to consecutive integers. This is illustrated by the "head injuries" data set. All the predictors had integer values like 1,2,3. Coding the missing information as 0 then gave a data set with predictors that were integer valued.

Grouping the Categories

The basic operation in modeling the effect of some predictor is the reduction of its values to different groups of categories. FIRM does this by testing whether the data corresponding to different categories of the dependent variable differ significantly, and if not, it puts these categories together. It does this differently for different types of predictors:

- *free* predictors are on the nominal scale. When categories are tested for possibly being together, any classes of a free predictor may be grouped. Thus in the car repairs, we could legitimately group any of the categories GM, Ford, Chrysler, Japanese and European together.
- *monotonic* predictors are on the ordinal scale. One example might be a rating with options "Poor", "Fair", "Good", "Very good" and "Excellent". Internally, these values need to be coded as consecutive integers. Thus in the "head injuries" data the different age groups were coded as 1, 2, 3, 4, 5, 6, 7 and 8. When the classes of a predictor are considered for grouping, only groupings of contiguous classes are allowed. For example, it would be permissible to pool by age into the groupings {1,2}, {3}, {4,5,6,7,8}, but not into {1,2}, {5,6}, {3,4,7,8}, which groups ages 3 and 4 with 7 and 8 while excluding the intermediate ages 4 and 5.

The term "monotonic" may be misleading as it suggests that the dependent variable has to either increase or decrease monotonically as the predictor increases. Though this is usual when you specify a predictor as monotonic, it is not necessarily so. FIRM fits "step functions", so it could be sensible to specify a predictor as monotonic if the dependent variable first increased with the predictor and then decreased, so long as the response was fairly smooth.

Consider, for example, using a person's age to predict physical strength. If all subjects were aged under, say, 18, then strength would be monotonically increasing with age. If all subjects were aged over 25, then strength would be monotonically decreasing with age. If the subjects span both age ranges, then the relationship would be initially increasing, then decreasing. Using age as a monotonic predictor will still work correctly in that only adjacent age ranges will be grouped together. The fitted means, though, would show that there were groups of matching average strength at opposite ends of the age spectrum.

If you have a predictor that is on the ordinal scale, but suspect that the response may not be smooth, then it would be better, at least initially, to specify it as free and then see if the grouping that came back looked reasonably monotonic. This would likely be done in the car repair costs example with the predictor *year*.

Additional Predictor Types

These are the two basic predictor types inherited from FIRM's roots in AID. In addition to these, FIRM has another three predictor types:

1. The *floating* predictor is a variant of the monotonic predictor. A floating predictor is one whose scale is monotonic except for a single "floating" class whose position in

the monotonic scale is unknown. This type is commonly used to handle a predictor which is monotonic but which has missing values on some cases. The rules of grouping are that the monotonic portion of the scale can be grouped only monotonically, but that the floating class may be grouped with any other class or classes on the scale. Like the free and monotonic predictors, this predictor type must also be coded in the data set as consecutive integers, and in these implementations of FIRM, the missing class is required to be the first one. So for example you might code a math grade of A, B, C, D, F or missing as 0 for missing, with A, B, C, D and F coded as 1, 2, 3, 4 and 5, for a total of 6 categories.

2. A *real* predictor is continuous: for example, the purchase price of the automobile; a patient's serum cholesterol level; a student's score in a pretest. FIRM handles real predictors by finding 9 cutpoints that will divide the range of values of the predictor into 10 classes of approximately equal frequency. Putting the predictor into one of these classes then gives a monotonic predictor. The real predictor may contain missing values. FIRM interprets any value in the data set that is not a valid number to be missing. For example, if a real predictor is being read from a data set and the value recorded is ? or . or *M*, then the predictor is taken to be missing on that case. Any real predictor that has missing values will be handled as a floating predictor—the missing category will be isolated as one category and the non-missing values will be broken down into the 10 approximately equal-frequency classes.
3. A *character* predictor is a free predictor that has not been coded into successive digits. For example, if gender is a predictor, it may be represented by M, F and ? in the original data base. To use it in FIRM, you could use some other software to recode the data base into, say, 1 for M, 2 for F, and 0 for ?, and treat it as a free predictor. Or you could just leave the values as is, and specify to FIRM that the predictor is of type character. FIRM will then find the distinct gender codes automatically.

   The character predictor type can also sometimes be useful when the predictor is integer- valued, but not as successive integers. For example, the number of cylinders of our hypothetical car, 4, 6, 8 or 12, would best be recoded as 1, 2, 3 and 4, but we could declare it character and let FIRM find out for itself what different values are present in the data. Releases of FIRM prior to 2.0 supported only free, monotonic and floating predictors. FIRM 2.0 and up implement the real and character types by internally recoding real predictors as either monotonic or floating predictors, and character as free predictors. Having a variable as type character does not remove the restriction on the maximum number of distinct values it can have—this remains at 16 for CATFIRM and (in the current release) 20 for CONFIRM. As of FIRM 2.0, the dependent variable in CATFIRM may also be of type character. With this option, CATFIRM can find out for itself what are the different values taken on by the dependent variable.

Missing Information and Floating Predictors

The monotonic and free predictor types date right back to the early 1960s implementation of AID; the real and character predictor types are essentially just an extra code layer on top of the basic types to make them more accessible. As the floating predictor type is newer, and also is not supported by all recursive modeling procedures, some comments on its potential uses may be helpful. The floating predictor type is mainly used for handling missing information in an otherwise ordinal predictor. Most procedures for handling missing information have at some level an underlying "missing at random" assumption. This assumption is not always realistic: there are occasions in which the fact that a predictor is missing in a case may itself convey strong information about the likely value of the dependent variable.

There are many examples of potentially informative missingness; we mention two. In the "head injuries" data, observations on the patient's eye function could not be made if the eye had been destroyed in the head injury. This fact may itself be an indicator of the severity of the injury. Thus, it does not seem reasonable to assume that the eye measurements are in any sense missing at random; rather it is a distinct possibility that missingness could predict a poor outcome. More generally in clinical contexts, clinicians tend to call for only tests that they think are needed. Thus, patients who are missing a particular diagnostic test do not have a random outcome on the missing test, but are biased towards those in whom the test would have indicated normal function.

Another example we have seen is in educational data, where Kass and Hawkins attempted to predict college statistics grades on the basis of high school scores. In their student pool, high school math was not required for college entry, and students who had not studied math in high school therefore had missing values for their high school math grade. But it was often the academically weaker students who chose not to study math in high school, and so students missing this grade had below average success rates in college. For the math grade to be missing was therefore in and of itself predictive of below-average performance.

FIRM's floating predictors provide a way to accommodate this predictive missingness. If a predictor really is missing at random, then the floating category will tend to be grouped with other categories near the middle of the ordinal part of the scale. But if the missingness is predictive, then the floating category will be either segregated out on its own or grouped with categories near one end of the ordinal part of the scale. In other words, predictors may be missing at random or in an informative way; FIRM can handle both situations, and the grouping of the floating category will provide a diagnosis of whether the missing information is random or informative. Inspection of the groupings produced by FIRM for the "head injuries" data does indeed suggest informative missingness on several predictors. For example *EMV* and *MRP*, with their 7-class monotonic plus floating class show this by the missing class being grouped with classes 6 and 7 on the scale.

No special predictor type is needed for missing information on a free predictor; all that is necessary is to have an extra class for the missing values. For example, if in a survey of adolescents you measured family type in four classes:

- 1: living with both birth parents;
- 2: with single parent;
- 3: with blended family;
- 4: with adoptive family;
- then respondents for whom the family type was not observed would define a fifth class, and the five classes would define a free (nominal) scale.

In the "head injuries" data set, *age* was always observed, and is clearly monotonic. *EMV*, *MRP*, *Change* and *Eye indicator* all have a monotonic scale but with missing information. Therefore, they are used as floating predictors. With *Pupils*, we have a choice. The outcome was binary, with missing information making a third category. Some thought shows that this case of three categories may be handled by making the predictor either free or floating—with either specification any of the categories may be grouped together and so we will get the same analysis.
We already mentioned that it is informative to see where the "missing information" category ends up in the grouping. Apart from the use mentioned of diagnosing whether missingness is predictive, this grouping may provide a useful fill-in for missing information on a predictor. For example, suppose the FIRM analysis groups the missing information category with categories 4, 5 and 6 of some floating predictor. If you are planning to do some other analyses, for example multiple regression, that cannot handle missing information, then it might be sensible to replace missing values on this predictor with one of the values 4, 5 or 6 for purposes of this other analysis. The justification for this substitution would be that, at least insofar as the dependent variable is concerned, cases missing this predictor behave like those with the predictor having values 4-6. The middle of this range, 5, is then a plausible fill-in value.

A second data set included in the **firmex** folder illustrates the use of character and real predictors. This data set (called **mixed.dat**) was randomly extracted from a much larger file supplied by Gordon V. Kass. It contains information on the high school records of students—their final scores in several subjects; when each wrote the University entrance examination and which of several possible examinations was taken; and the outcome of their first year at the university. This outcome is measured in two ways: the promotion code is a categorical measure of the student's overall achievement in relation to the requirements for going on to the second-year curriculum and can be analyzed using CATFIRM. The aggregate score is the average score received for all courses taken in the first University year, and is suitable for analysis using CONFIRM. Apart from having real and character predictors, this file also contains much missing information—for

example, the high school Latin scores are missing for the large number of students who did not study Latin in high school. Some of this missing information could be predictively missing.

Figure 3 shows the results of analyzing the promotion code using CATFIRM, while Figure 4 shows the results of analyzing the University aggregate score using CONFIRM.

Both dendrograms are smaller than those of the “head injuries” data set, reflecting the fact that there is much more random variability in the student’s outcomes than there is in the head injuries data. The FIRM models can isolate a number of significantly different groups, but there is substantial overlap in the range of performance of the different subgroups.

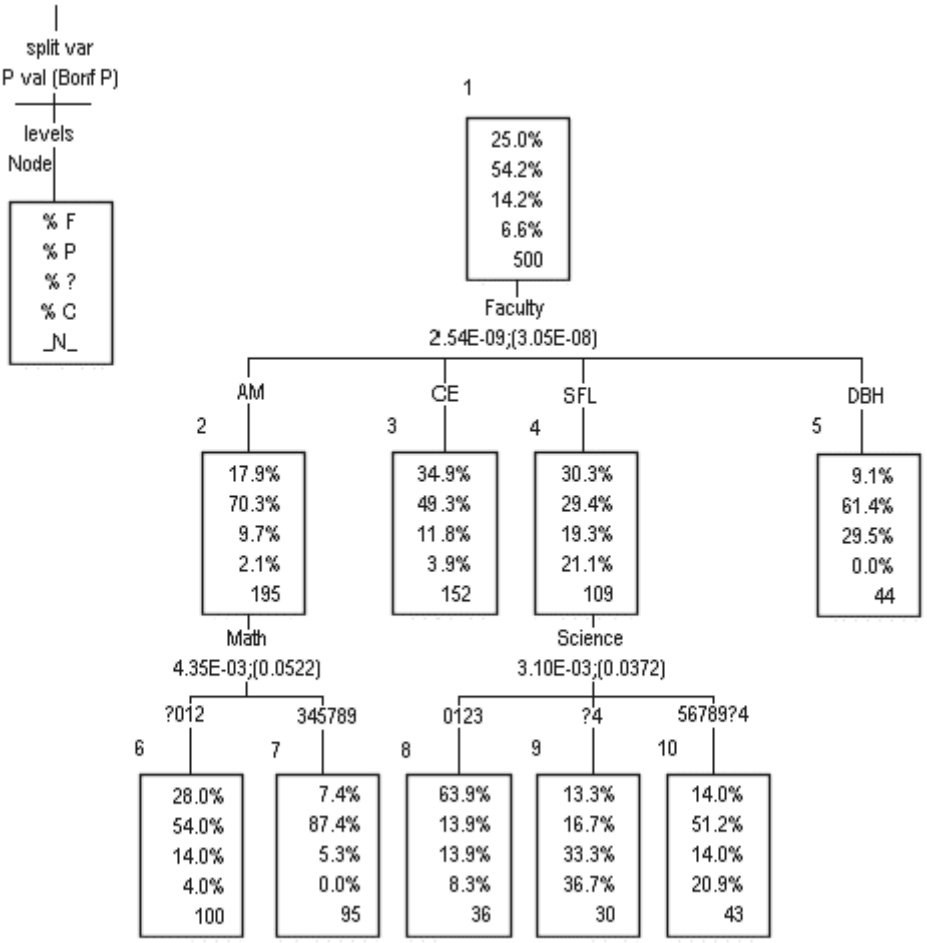

Figure 3: CATFIRM dendrogram of education data set

The CONFIRM analysis has terminal nodes ranging from a mean score of 42.2% up to 66.6%. This range is wide enough to be important academically (it corresponds to two letter grades), even though it is comparable with the within-node standard deviation of scores.

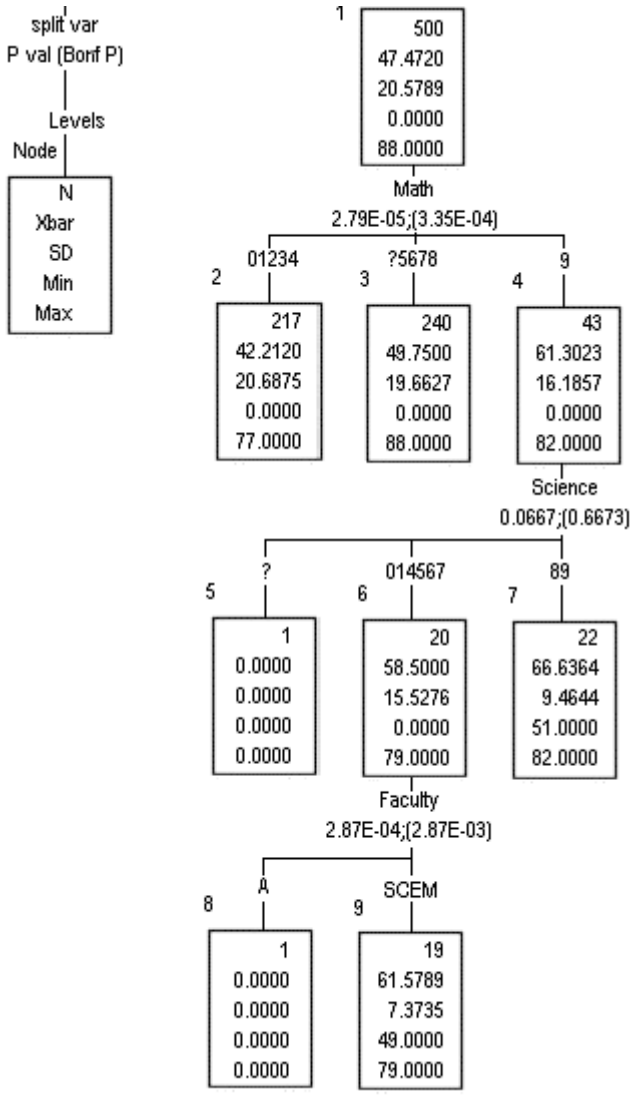

Figure 4: CONFIRM dendrogram of education data set

Outliers

The CONFIRM output also illustrates how FIRM reacts to outliers in the data. Nodes 5 and 8 in Figure 4 each contain a single student who dropped out and whose overall final score is recorded as zero—a value all but impossible for someone who was physically present at all final exams. FIRM discovers that it is able to make a statistically highly significant split on a predictor variable that happens to "fingerprint" this case, and it does so.

Overall Method of Operation

CONFIRM with a categorical (free) predictor is like a one-way analysis of variance on that predictor, except that CONFIRM groups together different predictor classes that seem not to differ significantly. CONFIRM for an ordinal predictor is a possible alternative to linear regression, with the difference that while linear regression assumes a constant rate of increase as you go from one level of the ordinal predictor to another, CONFIRM uses a "staircase" model where the dependent variable either remains constant as you go from one level to the next, or jumps by some (not necessarily constant) amount.

Most other recursive partitioning procedures fix the number of ways a node splits, commonly considering only binary splits. FIRM's method of reducing the classes of the predictors differs from these approaches. A *c*-category predictor may group into any of 1 through *c* categories, the software deciding what level of grouping the data appear to require. Both CONFIRM and CATFIRM follow the same overall approach. If a predictor has *c* classes, the cases are first split into *c* separate groups corresponding to these classes. Then tests are carried out to see whether these classes can be reduced to fewer classes by pooling classes pairwise. This is done by finding two-sample test statistics (Student's *t* for CONFIRM, chi-squared for CATFIRM) between each pair of classes that could legally be grouped together. The groups that can "legally" be joined depend on the predictor type:

- For a free predictor and a character predictor, any two groups can be joined.
- For a monotonic predictor, only adjacent groups may be joined.
- For a float predictor, any pair of adjacent groups can be joined. In addition, the floating category can join any other group.
- A real predictor works like a monotonic predictor if it has no missing information. If it is missing on some cases, then it works like a float predictor.

If the most similar pair fails to be significantly different at the user-selected significance level, then the two classes are merged into one composite class, reducing the number of groups by 1. The pairwise tests are then repeated for the reduced set of *c* - 1 classes. This process continues until no legally poolable pair of simple or composite classes is separated by a non-significant test statistic, ending the "merge" phase of the analysis. Let's illustrate this process with CONFIRM's handling of the floating predictor *EMV* in the "head injuries" data. The process starts with the following overall summary statistics for the grouping of the cases by *EMV*:

```
predictor no. 2 EMV
statistics before merging
Cate   ?    1    2    3    4    5    6    7
mean 2.07 1.00 1.19 1.58 1.81 1.84 2.25 2.63
size   28   19   64   52  111   96   65   65
s.d.  .94  .00  .53  .82  .93  .91  .94  .74
```

Any of the categories can merge with its immediate left or right neighbors, and in addition the float category can merge with any group. The "split" output file lists the following summary statistics. The first line of "merge stats" shows the Student's *t*-value for merging the two groups between which it lies; the second shows the Student's *t*-value distinguishing the float category from the category to the right. So, for example, the *t*-value testing whether we can merge categories 3 and 4 is 1.7; and the *t*-value for testing whether we can merge the float category with category 4 is -1.5.

```
Group      ?     1      2     3     4     5     6     7
merge stats    -4.3     .9   2.5   1.7    .3   3.0   2.6
                4.3   -4.7  -2.5  -1.5  -1.3    .9   3.0


Group      ?     1      2     3     45    6     7
merge stats    -4.3    .9    2.5   1.9   3.6   2.6
                4.3  -4.7   -2.5  -1.5    .9   3.0


Group      ?    12      3    45      6    7
merge stats    -5.1   2.9   1.9    3.6   2.6
                5.1  -2.5  -1.5     .9   3.0


Group           12      3    45    ?6      7
merge stats     2.9   1.9   3.5    3.3


Group          12     345     ?6     7
merge stats    6.0    4.1     3.2
```

The closest groups are 4 and 5, so these are merged to form a composite group. Once this is done, the closest groups are 1 and 2, so these are merged. Next ? and 6 are merged. Following this, there no longer is a floating category, so the second line of "merge stats" stops being produced. Finally, group 3 is merged with the composite 45. At this stage, the nearest groups (?6 and 7) are separated by a significant Student's *t*-value of 3.2, so the merging stops. The final grouping on this predictor is {12}, {345}, {?6}, {7}.

To protect against occasional bad groupings formed by this incremental approach, FIRM can test each composite class of three or more categories to see whether it can be resplit into two that are significantly different at the split significance level set for the run. If this occurs, then the composite group is split and FIRM repeats the merging tests for the new set of classes. This split test is not needed very often, and can be switched off to save execution time by specifying a split significance level that is smaller than the merge significance level.

The end result of this repeated testing is a grouping of cases by the predictor. All classes may be pooled into one, indicating that that predictor has no descriptive power in that node. Otherwise, the analysis ends with from 2 to *c* composite groups of classes, with no further merging or splitting possible without violating the significance levels set.

The last part of the FIRM analysis of a particular predictor is to find a formal significance level for it. In doing this, it is essential to use significance levels that reflect the grouping of categories that has occurred between the original *c*-way split and the final (say *k*-way) split. Hawkins and Kass (1982) mention two ways of measuring the overall significance of a predictor conservatively—the "Bonferroni" approach, and the "Multiple comparison" approach. Both compute an overall test statistic of the *k*-way classification: for CATFIRM a Pearson χ^2-statistic, and for CONFIRM a one-way analysis of variance F-ratio. The Bonferroni approach takes the *p*-value of the resulting test statistic and multiplies it by the number of implicit tests in the grouping from *c* categories to *k*. There are two possible multiple comparison *p*-values. The default in (versions of FIRM through FIRM 2.0) computes the *p*-value of the final grouping as if its test statistic had been based on a *c*-way classification of the cases. The default introduced in FIRM 2.1 is the *p*-value of the original *c*-way classification, but the earlier multiple comparison test can be selected if desired.

Since both the Bonferroni and multiple comparison approaches yield conservative bounds on the *p*-value, the smaller of the two values is taken to be the overall significance of the predictor.

The final stage is the decision of whether to split the node further and, if so, on which predictor. FIRM does this by finding the most significant predictor on the conservative

test, and making the split if its conservative significance level multiplied by the number of active predictors in the node is below the user-selected cutoff. The FIRM analysis stops when none of the nodes has a significant split.

Some users prefer not to split further any nodes that contain only a few, or very homogeneous, cases. Some wish to limit the analysis to a set maximum number of nodes. The codes contain options for these preferences, allowing the user to specify threshold sizes (both codes) and a threshold total sum of squares (CONFIRM) that a node must have to be considered for further splitting, and to set a maximum number of nodes to be analyzed.

Comparison with Other Codes

Before discussing the details of using the codes, we mention some historical development and other related approaches. The original source on recursive partitioning was the work of Morgan and Sonquist (1963) and their "Automatic Interaction Detector" (AID). This covered monotonic and free predictors, and made only binary splits. At each node, the split was made where the explained sum of squares was greatest, and the analysis terminated when all remaining nodes had sums of squared deviations below some user-set threshold.

It was soon found that the lack of a formal basis for stopping made the procedure prone to overfitting. In addition, the use of explained sum of squares without any modification for the number of starting classes or the freedom to pool them also made AID tend to prefer predictors with many categories to those with fewer, and to prefer free predictors to monotonic.

Breiman *et al* (1984) built on the AID foundation with their Classification and Regression Trees approach. This has two classes of predictors—categorical (corresponding to AID's "free" predictors) and continuous; unlike FIRM, their method does not first reduce continuous (in FIRM terminology "real") predictors down to a smaller number of classes, nor does it draw any distinction between these and monotonic predictors with just a few values. The Breiman *et al* methodology does not have an equivalent to our "floating" category. Instead, missing values of predictors are handled by "surrogate splits", by which when a predictor is missing on some case, other predictors are used in its stead. This approach depends on the predictive power of the missingness being captured in other non-missing predictors.

Breiman *et al* use only binary splits. Where FIRM chooses between different predictors on the basis of a formal test statistic for identity, they use a measure of "node purity", with the predictor giving the purest descendant nodes being the one considered for use in

splitting. This leads to a tendency to prefer categorical predictors to continuous, and to prefer predictors with many distinct values to predictors with few. FIRM's use of conservative testing tends to lead to the opposite biases, since having more categories, or being free rather than monotonic, increases the Bonferroni multipliers applied to the predictor's raw significance level.

With a continuous dependent variable, their measure of node purity comes down to essentially a two-sample *t* (which FIRM would also use for a binary split), but their purity measure for a categorical dependent variable is very different than FIRM's. FIRM looks for statistically significant differences between groups while they look for splits that will classify to different classes. So the Breiman *et al* approach seeks two-column cross classifications with different modal classes—that is, in which one category of the dependent variable has the largest frequency in one class while a different category has the largest frequency in the other class. FIRM looks instead for groupings that give large chi-squared values, which will not necessarily have different modal classes. This important difference has implications for the use of the two procedures in minimal-model classification problems. As it is looking for strong relationships rather than good classification rules, a FIRM analysis may produce a highly significant split, but into a set of nodes all of which are modal for the same class of the dependent variable.

An example of this is the split of Node 4 of the head injuries data set (Figure 2). This creates two descendant nodes with 52% and 86% moderate to good recoveries. If all you care about is predicting whether a patient will make a moderate to good recovery, this split is pointless—in both descendant nodes you would predict a moderate to good recovery. But if you want to explore the relationship between the indicator and the outcome, this split would be valuable, as the proportions of favorable outcomes are substantially different.

It might also be that a clinician might treat two patients differently if it is known one of them has a 86% chance of recovery while the other has a 52% chance, even though both are expected to recover. In this case, too, the FIRM split of the node could provide useful information.

Two major differences between FIRM and the Classification and Regression Trees method relate to the rules that are used to decide on the final size of tree. FIRM creates its trees by "forward selection". This means that as soon as the algorithm is unable to find a node that can be split in a statistically significant way, the analysis stops. Since it is possible for a non-explanatory split to be needed before a highly explanatory one can be found at a lower level, it is possible for the FIRM analysis to stop too soon and fail to find all the explanatory power in the predictors. It is easy to construct artificial data sets in which this happens.

You can protect yourself against this eventuality by initially running FIRM with a lax splitting criterion to see if any highly significant splits are hidden below non-significant ones; if not, then just prune away the unwanted non-significant branches of the dendrogram. Particularly when using the printed dendrogram, this is a very easy operation as all the splits are labeled with their conservative significance level. We do not therefore see this theoretical possibility of stopping too soon as a major practical weakness of FIRM's forward testing approach.

Breiman *et al* use the opposite strategy of "backward elimination". First, a deliberately very oversized tree is created by continuing to split nodes, whether the splits produce some immediate benefit or not. Then the oversized tree is pruned from the bottom, undoing splits that appear not to correspond to distinct patterns in the data.

The second, more profound difference is in the procedure used to decide whether a particular split is real, or simply a result of random sampling fluctuations giving the appearance of an explanatory split. FIRM addresses this issue using the Neyman Pearson approach to hypothesis testing. At each node, there is a null hypothesis that the node is homogeneous, and this is tested using a controlled, conservative significance level. In the most widely-used option, Breiman *et al* decide on the value of a split using cross validation. We will do no more than sketch the implications, as the method is complicated (see for example Breiman, *et al*, 1984 or McLachlan, 1992). An original very oversized tree is formed. The sample is then repeatedly (typically 10 times) split into a "calibration" subsample and a "holdback" sample. The same tree analysis is repeated on the calibration subsample as was applied to the full data and this tree is then applied to the holdback sample, to get an "honest" measure of its error rate. The picture obtained from these 10 measures is then used to decide how much of the detail of the original tree was real and how much due just to random sampling effects. This leads to a pruning of the original tree, cutting away all detail that fails to survive the cross-validation test.

As this sketch should make clear, an analysis using the Breiman *et al* methodology generally involves far more computing than a FIRM analysis of the same data set. As regards stopping rules, practical experience with the two approaches used for a continuous dependent variable seems to be that in data sets with strong relationships, they tend to produce trees of similar size and form. In data sets with weak, albeit highly significant relationships, FIRM's trees appear to be generally larger and more detailed. See Hawkins and McKenzie (1995) for some further examples and discussion of relative tree sizes of the two methods.

In view of the very different objectives of the two approaches to analysis of categorical data (getting classification rules and uncovering relationships respectively), there are probably few useful general truths about the relative performance of the two approaches for a categorical dependent variable.

Another recursive partitioning method is the Fast Algorithm for Classification Trees (FACT), of Loh and Vanichsetakul (1988). This method was designed for a categorical dependent variable, and treats the splitting by using a discriminant analysis paradigm.

The KnowledgeSEEKER procedure is also based on the CHAID algorithm and produces results similar to those of CATFIRM. The main technical difference from FIRM (other than the more user-friendly interface) lies in the conservative bounds used to assess a predictor, where KnowledgeSEEKER uses a smaller Bonferroni multiplier than does FIRM.

The public domain SAS procedure TREEDISC (available on request by Warren Sarle of SAS Institute) also implements an inference-based approach to recursive partitioning. The SAS Data Mining module also contains a variety of RP approaches including all three widely-used approaches.

This list is far from exhaustive—there are many other RP approaches in the artificial intelligence arena—but perhaps covers some of the major distinct approaches.

References

Breiman, L., Friedman, J.H., Olshen, R.A., and Stone, C.J. (1984). *Classification and Regression Trees*. Wadsworth: Belmont.

Hawkins, D.M., and Kass, G.V. (1982). Automatic Interaction Detection. In: Hawkins, D.M. (Ed.) *Topics in Applied Multivariate Analysis*, Cambridge University Press: Cambridge.

Hawkins, D.M., and McKenzie, D.P. (1995). *A data-based comparison of some recursive partitioning procedures*. In: Proceedings, Statistical Computing Section, American Statistical Association, 245-252.

Hawkins, D.M., Young, S.S., and Rusinko, A. (1997). Analysis of a large structure-activity data set using recursive partitioning. To appear in *Quantitative Structure Activity Relationships*.

Hooton, T.M., Haley, R.W., Culver, D.H., White, J.W., Morgan, W.M., and Carroll, R.J. (1981). The joint associations of multiple risk factors with the occurrence of nosocomial infections. *American Journal of Medicine*, **70**, 960-970.

Kass, G.V. (1975). *Significance testing in, and an extension to Automatic Interaction Detection*. Ph.D. Thesis, University of the Witwatersrand: Johannesburg.

Kass, G.V. (1980). An exploratory technique for investigating large quantities of categorical data. *Applied Statistics*, **29**, 119-127.

Loh, W-Y., and Vanichsetakul, N. (1988). Tree structured classification via generalized discriminant analysis. *Journal of the American Statistical Association*, **83**, 715-725.

McLachlan, G.J. (1992). *Discriminant Analysis and Statistical Pattern Recognition*. Wiley: New York.

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

Titterington, D.M., Murray, G.D., Murray, L.S., Spiegelhalter, D.J., Skene, A.M., Habbema, J.D.F., and Gelpke, G.J. (1981). Comparison of discrimination techniques applied to a complex data set of head injured patients. *Journal of the Royal Statistical Society*, **A**, 144, 145-161.