Paper AA-42

An introduction to classification and regression trees with PROC HPSPLIT

Peter L. Flom Peter Flom Consulting, LLC

ABSTRACT

Classification and regression trees are extremely intuitive to read and can offer insights into the relationships among the IVs and the DV that are hard to capture in other methods. I will introduce these methods and illustrate their use with PROC HPSPLIT.

Keywords: Trees.

INTRODUCTION

Regression and classification trees are methods for analyzing how one dependent variable (DV) is related to multiple independent variables (IV). Regression trees deal with continuous DV and classification trees deal with categorical ones. The essential idea of both is fairly simple: We start with all the observations in one big group. We split it into two groups by choosing the best value of any of the IV for the split, this gives two child nodes. Then we repeat this process. There are numerous ways to define 'best' and there are also numerous ways of figuring out how big the tree should be. The full tree is usually too detailed and overfits ths data. The final tree is pruned in an attempt to avoid this.

ADVANTAGES AND DISADVANTAGES OF TREES

Advantages of trees include:

- Easy to interpret
- Can find interactions in interesting ways
- Can deal with categorical and continuous IVs
- Do not require a lot of preprocessing
- Make no assumptions about the data
- Deal well with colinearity
- Are somewhat similar to human decision making

Disadvantages include:

- Can be non-robust, especially in small datasets
- It is not possible to guarantee the 'best' tree - the problem is NP complete
- Can sometimes overfit (but this can sometimes be dealt with)

THEORY

BUILDING THE TREE

There are various methods of splitting each node. For classification trees, the default is entropy. For regression trees it is restricted sum of squares (RSS). You can set a maximum depth for the tree (the default is 10). Especially when you have a lot of data (with 'a lot' being machine dependent) `PROC HPSPLIT` can be time consuming. You can set a lower limit when exploring, just to see the first few splits. The vast majority of trees use two branches for each split. `PROC HPSPLIT` does allow you to use more branches per split with `MAXBRANCH`.

PRUNING THE TREE

Once the full tree is grown, it must be pruned to avoid overfitting (one exception would be if you set a maximum depth that was smaller than the full tree and that no pruning was then needed). The default method is cost complexity. Another method is to look at the nodes and decide based on substantive knowledge which splits are not needed.

PROC HPSPLIT

Here I outline the basic syntax of PROC HPSPLIT and do not go over every detail. For that, you can always see the documentation.

```
PROC HPSPLIT <options>;
CLASS variable... </options>;
CODE FILE=filename;
GROW criterion </ options>;
ID variables;
MODEL response <(response-options)> = variable <variable...>;
OUTPUT output-options;
PARTITION <partition-options>;
PERFORMANCE performance-options;
PRUNE prune-method <(prune-options)>;
RULES FILE=filename;
```

PROCHPSPLIT starts the procedure. It and MODEL are required. Key and uncommon options on PROC HPSPLIT include

- NODES which prints a table of each node of the tree with associated information,
- ASSIGNMISSING which deals with missing values
- MAXDEPTH which sets the maximum depth of the tree
- MAXBRANCH which sets the maximum number of branches at each split in the tree

The CLASS, MODEL, ID and OUTPUT statements work in more or less familiar ways, although there are some different options. CODE generates a SAS® program file that can score new data sets, PRUNE and GROW allow you to set methods for growing and pruning the tree.

REGRESSION TREE EXAMPLE: BIRTHWEIGHT DATA

Predicting low birth weight is important because babies born at low weight are much more likely to have health complications than babies of more typical weight. The usual approaches to this are either to model the mean birth weight as a function of various factors using OLS regression, or to dichotomize or otherwise categorize birth weight and then use some form of logistic regression (either 'normal' or ordinal). Both these are inadequate. Modeling the mean is inadequate because, as we shall see (in the paper on QUANTREG), different factors are important in modeling the mean and the lower quantiles; this difficulty will be reflected in the rather poor results, below. We are often interested in predicting which mothers are likely to have the lowest weight babies, not the average birth weight of a particular group of mothers. Categorizing the dependent variable is rarely a good idea, principally because it throws away useful information and treats people within categories as the same. A typical cutoff value for low birth weight is 2.5 kg. Yet this implies that a baby born at 2.49 kg is the same as a baby born at 1.0 kg, while one born at 2.51 kg is the same as one who is 4 kg. This is clearly not the case. In addition, trees do not work well on highly imbalanced data.

SAS provides birthweight data that is useful for illustrating PROC HPSPLIT. I added an ID variable to the data set provided by SAS (this will be useful later):

```
data new;
  set sashelp.bweight;
  count + 1;
 run;
```

Then running the basic HPSPLIT is fairly straightforward:

```
proc hpsplit data=new seed=123;
 class black boy married momedlevel momsmoke ;
 model weight = black boy married momedlevel momsmoke momage momwtgain visit cigsperday;
 output out=bwregout;
run;
```

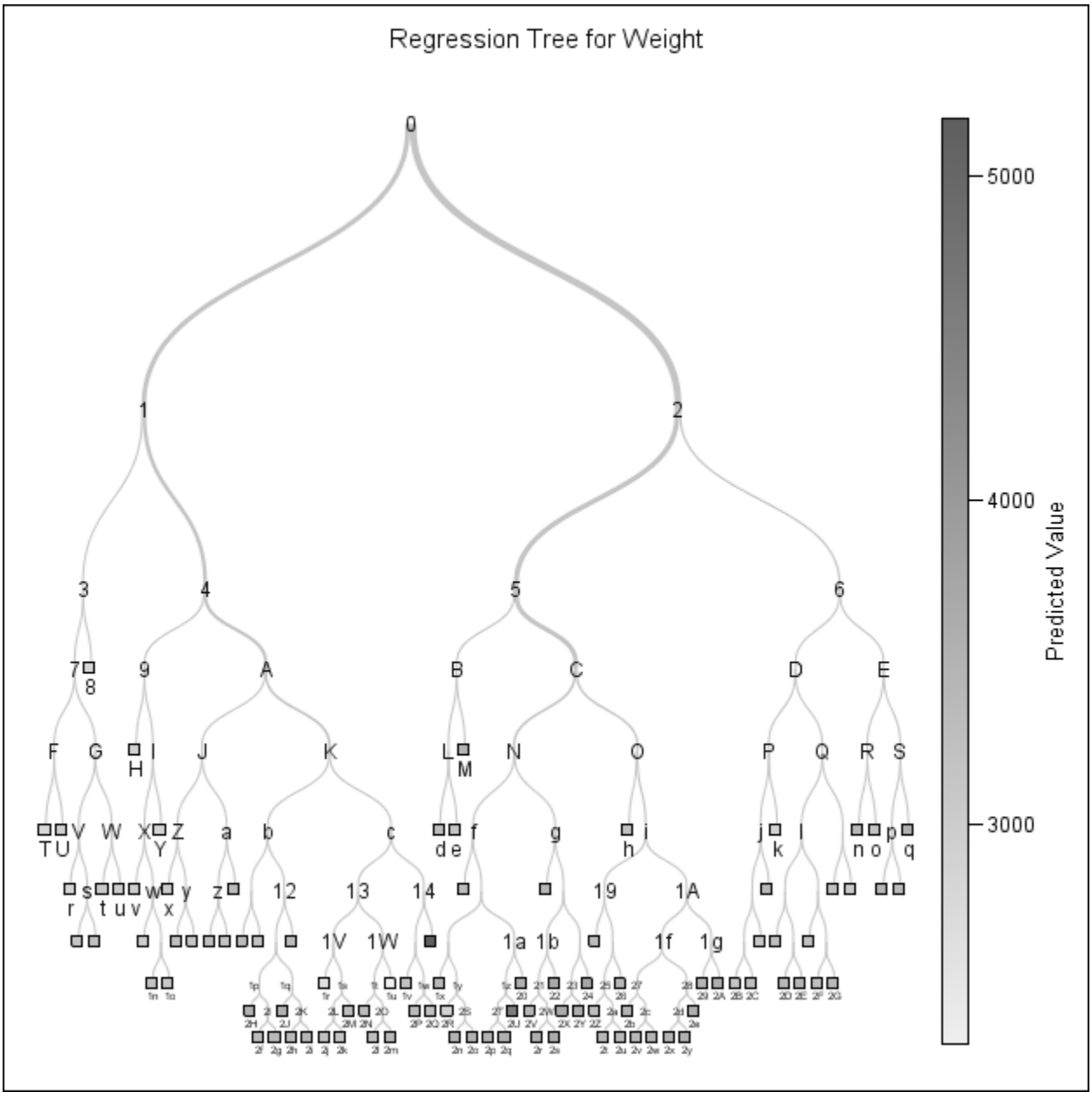

Figure 1: Whole tree plot of birth weight data

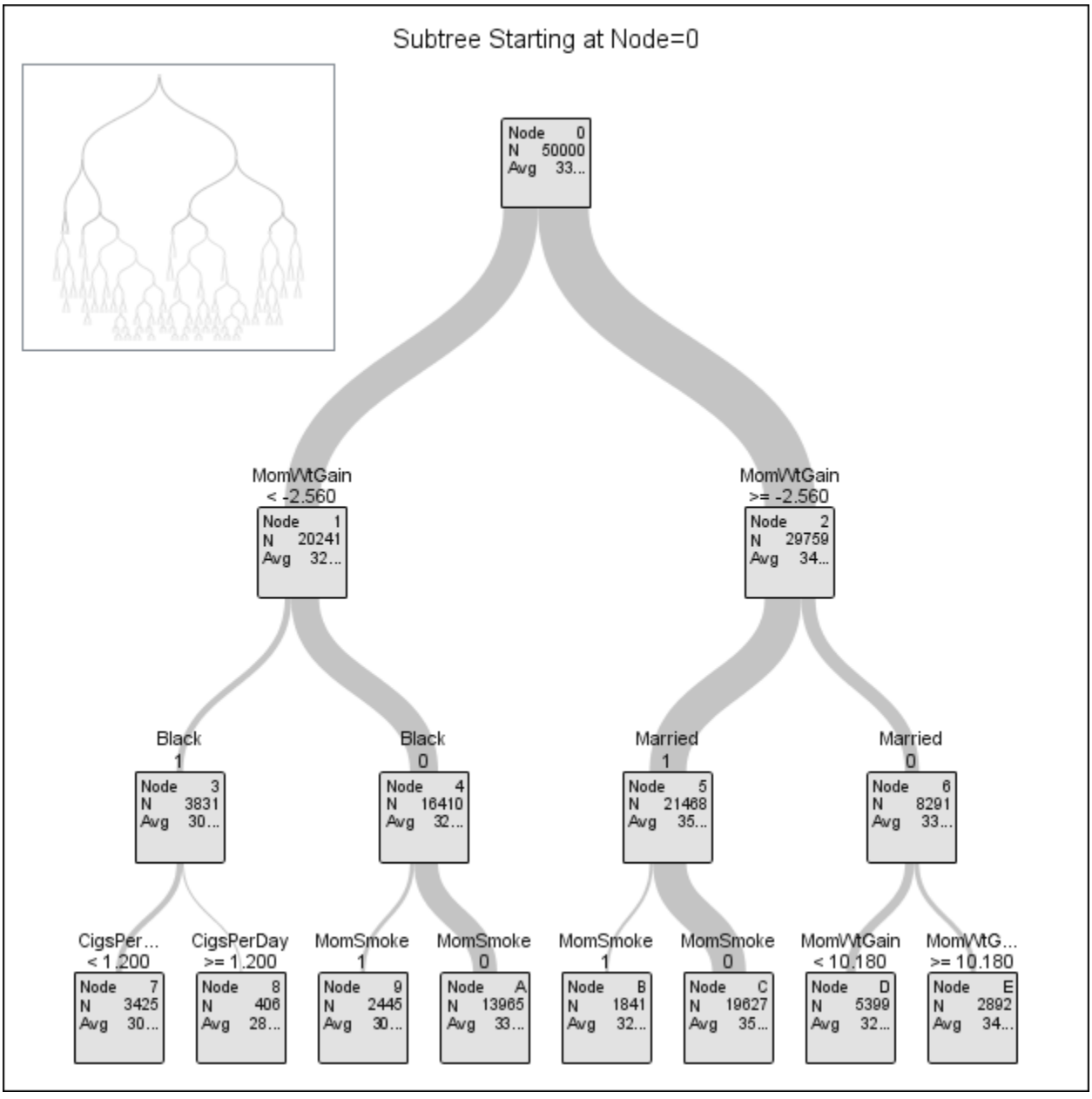

Figure 2: Zoomed tree plot of main node birth weight data

which produces figures 1 and 2.

In this case, the first plot offers somewhat limited information; it is mostly useful as showing a full picture of the tree and for reference in later exploration. The second plot, though, is relatively straightforward to read. At the top, we have 5,000 observations in one node. The best split (for the default choices of best) is whether the mother's weight gain was more or less than 2.560 kg. If it is less, than the woman is in the left node with 20,241 women, if it is more, she is in the right node. Among the women who gained less weight, the best split is now whether the woman was Black or not. However, among the women who gained more weight, the best split is now whether she was married. This already illustrates a power of trees: This sort of interaction is impossible to model in regular regression. We can read down the rest of the tree in a similar manner. The default output of HPSPLIT, however, is a bit problematic in that we cannot see the mean in each node in detail. The way around this is two undocumented options in HPSPLIT: FRACPREC and PREDICTORPREC (see below), which seem to work sometimes and not others. In addition, it can be useful to see a table of the final nodes and some information about them; this is done with NODES (in this case, the NODES results are very lengthy).

```
proc hpsplit data=new seed=123 plots = zoomedtree(nodes = ("0") depth = 2 fracprec = 4
      predictorprec = 4) nodes;
   class black boy married momedlevel momsmoke ;
   model weight = black boy married momedlevel momsmoke momage momwtgain visit cigsperday;
   output out=bwregout;
run;
```

which produces figure 3 where each node is labeled with the sample size and the mean birth weight of the babies in that node.

If you want to look in more detail at particular areas of the tree, you can use the PLOTS = ZOOMEDTREE option. From the larger tree, we can see that node 8 is a final node and needs no more exploration, but we can look at detail of node 9, for example, with .

```
 proc hpsplit data=new seed=123
   plots = zoomedtree(nodes = ("9") depth = 4 fracprec = 4 predictorprec = 4) nodes;
   class black boy married momedlevel momsmoke ;
   model weight = black boy married momedlevel momsmoke momage momwtgain visit cigsperday;
run;
```

which yields figure 4.

We can similarly explore other nodes. I suggest including multiple nodes in a single run of HPSPLIT, to save time.

We can also look at the means, standard deviations and so on, using the output data. A simple table from PROC MEANS, however, is not very informative. Every node (except one very tiny one) has a minimum that is dangerously low. But we can look at percentiles. If we found a node where a high proportion of babies were very small, then women in that node might be watched more carefully. If we run:

```
proc univariate data = bwregout;
  class _node_;
   var weight;
  output out = bwuni p1 = p1 p5 = p5 p10 = p10;
run;

proc sort data = bwuni;
 by p10;
run;

proc print data = bwuni;
run;
```

we can see that there are 19 nodes where more than 10% of babies are under 2500g. None of these nodes are particularly large, but they may be useful.

One issue is that the node labeling in the tree itself uses base 62 (0-9, A-Z, a - z). This makes for shorter labels on the tree, but necessitates a recoding for the tables. You can convert them in SAS, with a program I adapted from the SAS site.

```
data tree.bwregout;
```

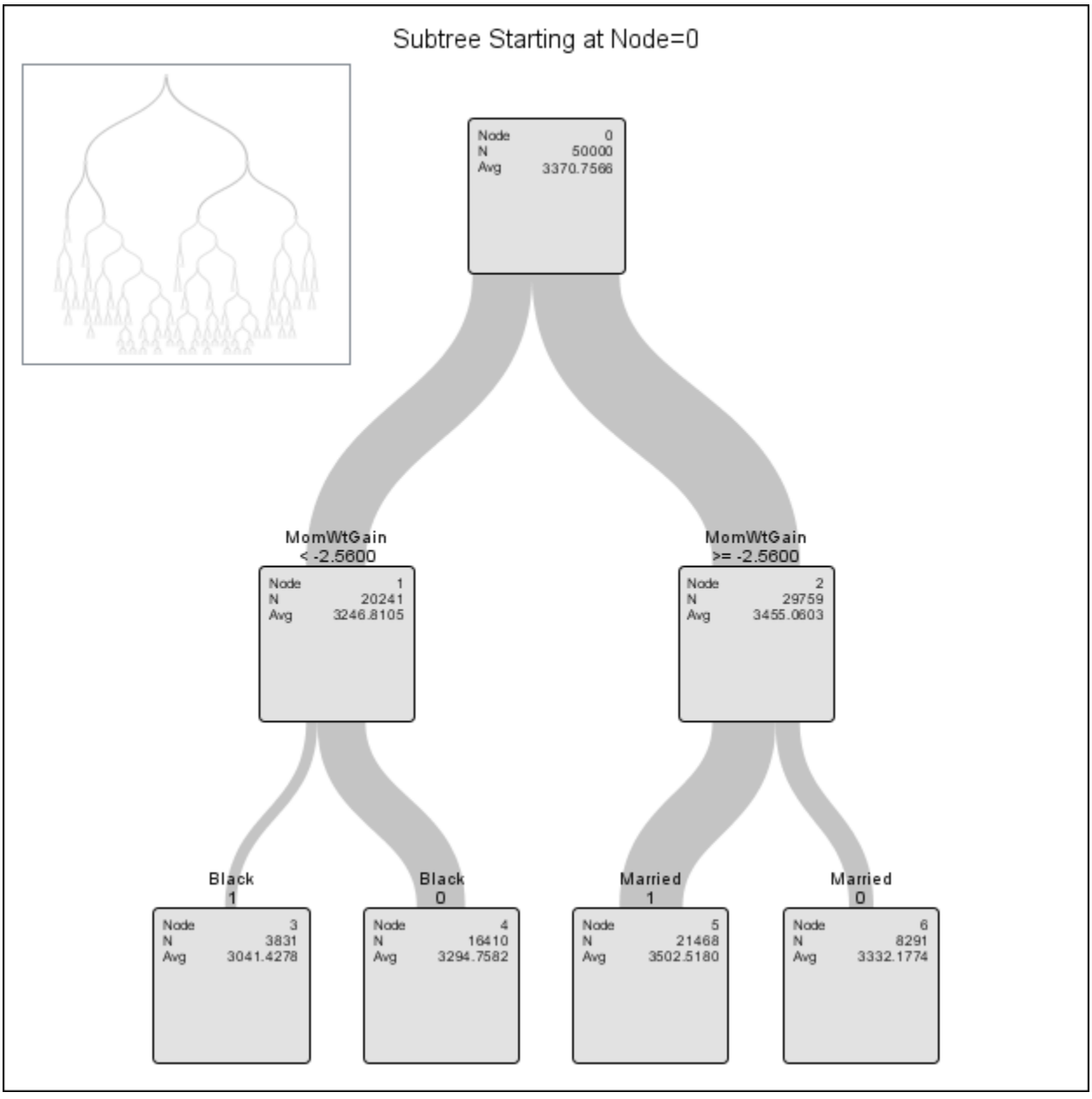

Figure 3: Zoomed tree plot of main node birth weight data

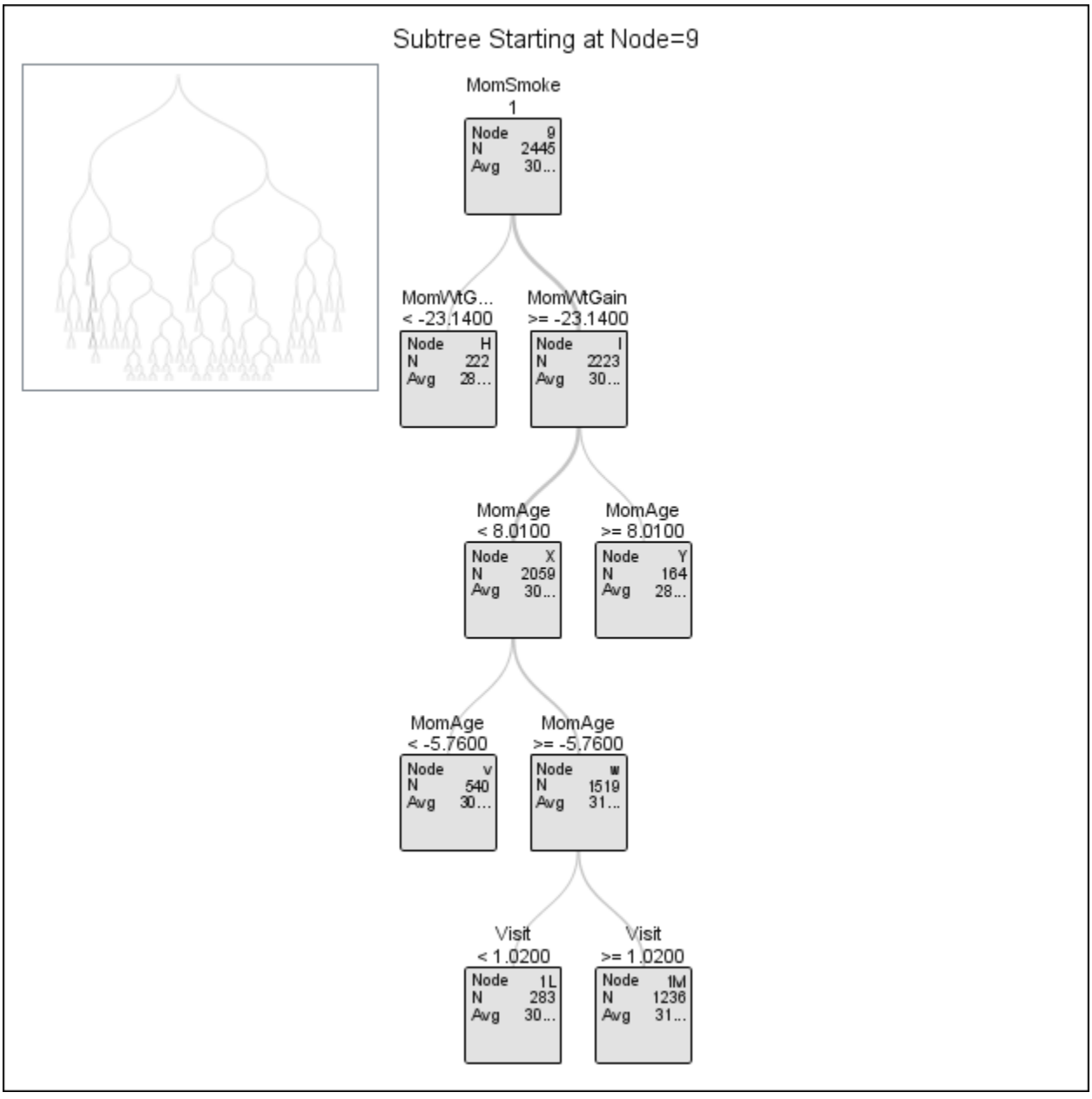

Figure 4: Zoomed tree plot of main node birth weight data

```
 set tree.bwregout;
 retain possdigs '0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz';
   do power=0 to 100 while(62**power<=_NODE_);  end;
   left=_NODE_;
   length node $10;
   node='␣';
   i=0;
   do power=power-1 to 0 by -1;
      i+1;
      r=int(left/(62**power));
      substr(node,i,1)=substr(possdigs,r+1,1);
      left=left-(62**power)*r;
   end;
run;
```

CLASSIFICATION TREE EXAMPLE: BIRTHWEIGHT DATA

Suppose that, against your advice, you are required by your 'pointy haired boss' (could be an actual boss, a professor, a journal editor or a colleague) to classify birth weight. One PHB wants two categories (low and normal) and the other wants three (very low, low and normal). We can create these easily in a data step and then run a tree on that. One note: We have to include the dependent variable on the CLASS statement. First we'll try the two category model:

```
proc hpsplit data = new seed = 123;
   class black boy married momedlevel momsmoke bwcat;
   model bwcat2 = black boy married momedlevel momsmoke momage momwtgain visit cigsperday;
   output out=hpsplout;
run;
```

This tree does not work very well, mostly because the data set is so unbalanced. The sensitivity is 0.05 while the specificity is 0.998. We can, however, adjust this:

```
data bwpred (keep = actual predicted);
  set new;
  %include "bw2catscore.sas";
  actual = bwcat2;
  predicted = (P_bwcat2Low > 0.05);
run;

proc freq data = bwpred;
 table actual*predicted;
run;
```

and this yields more balanced sensitivity and specificity.

However, when we try to run a tree on the three category variable, we get a very bad classification. One possible solution to this would be to run two trees: The first would be as above, separating the whole data set into two groups, then a second tree would be run on just the low birthweight babies. The top part of this tree is shown in figure 5.

As expected, this tree also has a problem because of the imbalance (only 8% of the very low BW babies are classified correctly), but we can fix that as above. When we do this, about 65% of babies are correctly classified.

REGRESSION TREE EXAMPLE: GAS MILEAGE

Kuhfeld & Cai (2013) analyzed automobile mileage data provided by the University of California at Irvine Machine Learning Data Depository (Asuncion & Newman, 2007). I provide a slightly different and further analysis of that data set. The first step is to get the data:

```
%let base = http://archive.ics.uci.edu/ml/machine-learning-databases;
```

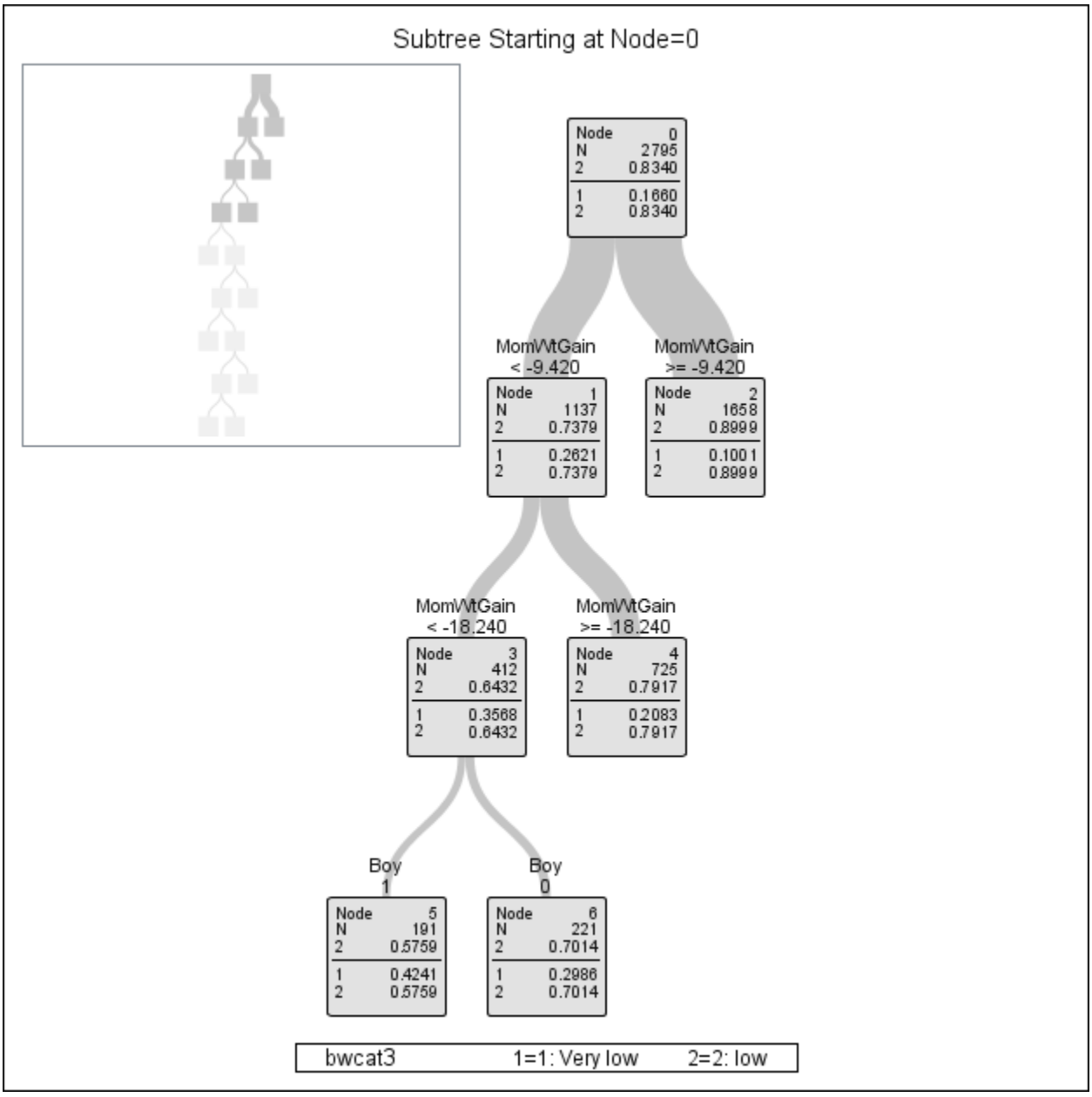

Figure 5: Plot of main node birth weight data low vs. very low

```
proc format;
  invalue q '?' = .;
run;

data autoMPG;
 infile "&base/auto-mpg/auto-mpg.data" device = url expandtabs lrecl = 120 pad;
 input MPG Cylinders Displacement HP :q. weight acceleration year origin name $50.;
 name = propcase(left(tranwrd(name, '"', '')));
run;
```

Initial analysis showed that there were a very few cars with either 3 or 5 cylinders while the vast majority had an even number; in addition, these cars with an odd number of cylinders appeared different from the other in terms of gas mileage; they were deleted from the data for all further analyses. We can create a tree with this code:

```
proc hpsplit data = autompg2 seed = 123 plots = zoomedtree(nodes = ("0") depth = 4
     fracprec = 2 predictorprec = 2);
  class origin cylinders;
 model mpg = cylinders displacement weight acceleration year origin;
   output out=mpgout;
    rules file = "\path\mpgrule.txt";
run;
```

For viewing the tree itself, there are two options: If we use depth = 4, then we see the whole tree but lose some of the information in each node. If we use depth = 2, then we need to use more than one plot. With depth = 4, we get figure **??**.

We can also get an analysis of how well the final nodes work:

```
proc means data = mpgout;
 class _NODE_;
 var MPG;
run;
```

which produces the table 1.

Table 1: Analysis of nodes of MPG tree

Analysis Variable : MPG						
Node number	N Obs	N	Mean	Std Dev	Minimum	Maximum
7	69	69	18.80	2.05	15.00	24.00
8	20	20	24.08	5.18	17.60	38.00
9	84	84	14.01	1.87	9.00	18.00
10	14	14	18.87	2.88	15.50	26.60
11	46	46	29.12	2.93	24.00	36.00
12	48	48	36.24	4.10	29.50	46.60
13	64	64	23.84	2.65	18.00	30.90
15	18	18	33.25	3.57	27.20	43.40
16	28	28	27.55	2.70	22.30	32.90

This tree works quite well. None of the final nodes are tiny and the sds in most of the nodes are small.

One interesting point is that some of the nodes are quite similar. For instance, nodes 7 and 10; note that in the figure, node 10 is node A. Both of these nodes contain cars that have 6 or 8 cylinders, but node 10 has larger displacement but neweer cars than node 7.

FIT STATISTICS

For a regression tree, SAS produces average square error (ASE) and residual sum of squares (RSS). These are not completely intuitive - they may be useful for comparing trees, but they lack an obvious interpretation. I think a useful method is to make a box plot with each node and the whole data set:

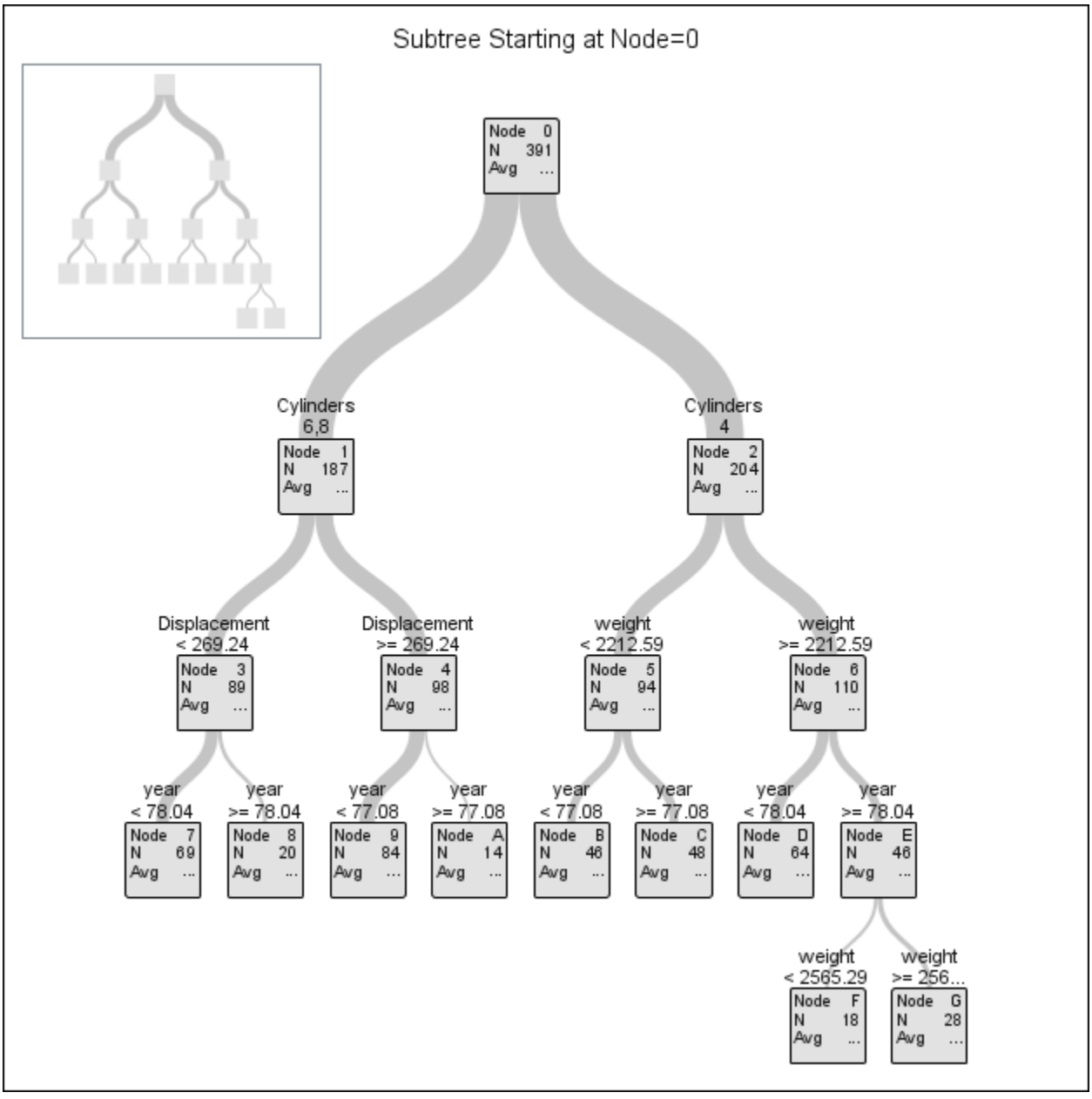

Figure 6: Whole tree plot of gas mileage data

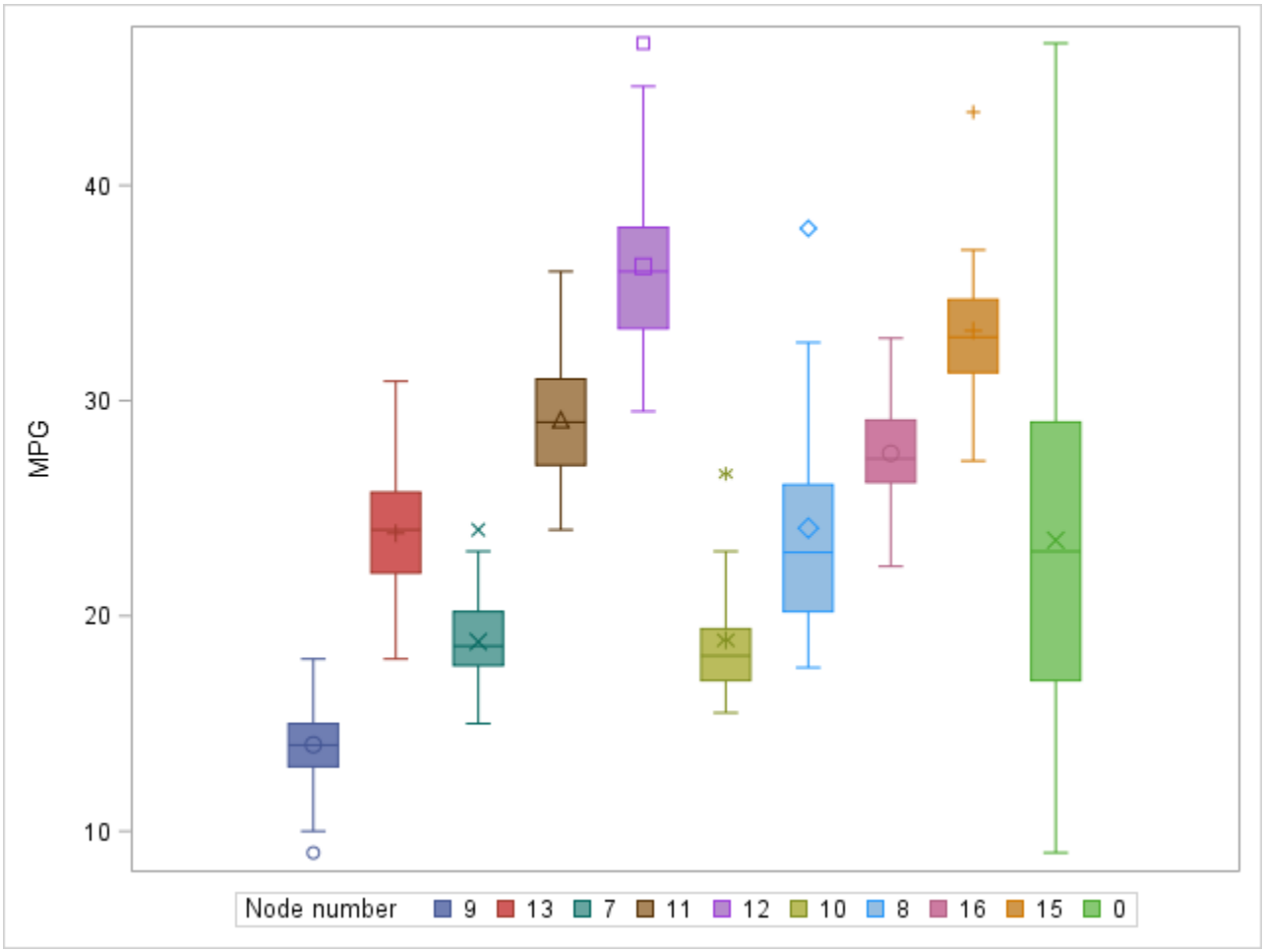

Figure 7: Whole tree plot of gas mileage data

```
data tree.vboxplot;
 set tree.mpgout
     tree.mpgout (in = in2);
 if in2 then _node_ = 0;
run;

proc sgplot data = tree.vboxplot;
 vbox mpg /group = _node_;
run;
```

which produces figure 7

where we can see that each node covers a much smaller range than the data.

SUMMARY AND CONTACT INFO

SUMMARY

PROC HPSPLIT is a good implementation of regression and classification trees, which can be very useful in some situations, are easy to interpret, and can provide insights not easily available through other methods.

CONTACT INFORMATION

Peter L. Flom
Peter Flom Consulting, LLC
515 West End Ave.

New York, NY 10024
Phone: (917) 488-7176
peterflomconsulting@mindspring.com
Personal webpage: http://www.statisticalanalysisconsulting.com/