partDSA: Deletion/Substitution/Addition Algorithm for Partitioning the Covariate Space in Prediction

Annette M. Molinaro
University of California, San Francisco

Stephen Weston
Yale University

Abstract

The **partDSA** package (Molinaro, Lostritto, and Weston 2009) provides a novel recursive partitioning tool for prediction when numerous variables jointly affect the outcome. In such settings, piecewise constant estimation provides an intuitive approach by elucidating interactions and correlation patterns in addition to main effects. As well as generating *'and'* statements similar to previously described methods, **partDSA** explores and chooses the best among all possible *'or'* statements. The immediate benefit of **partDSA** is the ability to build a parsimonious model with *'and'* and *'or'* conjunctions. Currently, **partDSA** is capable of handling categorical and continuous explanatory variables and outcomes. This vignette provides a guide for analysis with the **partDSA** package while the actual algorithm is introduced and thoroughly described in Molinaro, Lostritto, and van der Laan (2010).

Keywords: Recursive partitioning.

1. Introduction

Classification and Regression Trees (CART) (Breiman, Friedman, Olshen, and Stone 1984), a binary recursive partitioning algorithm, allows one to explore the individual contributions of various covariates as well as their interactions for the purposes of predicting outcomes. The end product of CART is a list of *'and'* statements. For example, a CART tree is illustrated in Figure 1. In this tree there are two variables (diagnosis age, a continuous variable, and tumor grade, an ordered variable) and the outcome is number of positive lymph nodes, a continuous variable. Thus, the parameter of interest is the conditional mean of the number of positive nodes given the covariates and the chosen loss function is the squared error. The splitting and pruning (details can be found in Breiman *et al.* (1984)) are based on the L_2 loss function and result in a tree with three terminal nodes. The final predictor can be read as: if a patient is less than 50 years of age, her predicted number of positive nodes is β_1; if a patient is over 50 *and* her tumor grade is less than or equal to 2, her predicted number of positive nodes is β_2; if a patient is over 50 *and* her tumor grade is greater than 2, her predicted number of positive nodes is β_3.

Although the importance of such statements is not in debate, there are settings where the *'and'* statement ordering does not accurately account for all of the observed biological phenomena.
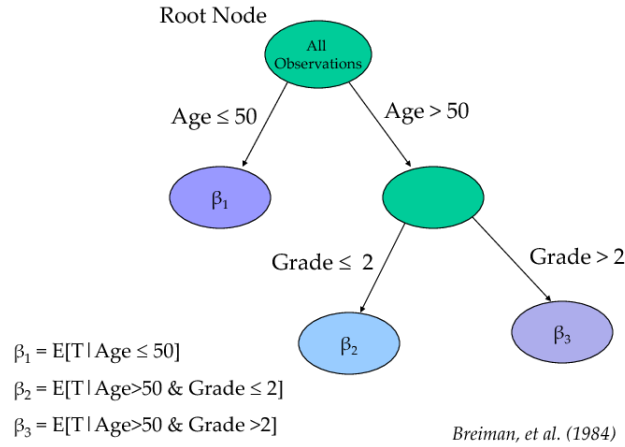
Figure 1: *Classification and Regression Tree Example.* This tree shows an example of recursive binary partitioning using CART with the variables 'Diagnosis Age' and 'Tumor Grade'. The terminal nodes have predicted values $\beta_1, \beta_2,$ and, β_3.

Figure 2 illustrates the scenario when the number of positive nodes is similar for women under 50 years of age and those over 50 whose tumor grade is less than or equal to 2, i.e. $\beta_1 = \beta_2$. The resulting model can be written as two partitions including an *'or'* statement as opposed to three terminal nodes consisting of three individual *'and'* statements.

The final predictor can be read as: if a patient is less than 50 years of age **OR** if a patient is over 50 *and* her tumor grade is less than or equal to 2, her predicted number of positive nodes is β_1; if a patient is over 50 *and* her tumor grade is greater than 2, her predicted number of positive nodes is β_2. This model can be drawn as shown in Figure 3. Thus, as well as generating *'and'* statements similar to CART, **partDSA** explores and chooses the best among all possible *'or'* statements. The immediate benefit of **partDSA** is the ability to build a stable and parsimonious model with *'and'* and *'or'* conjunctions.

To illustrate **partDSA**, in Section 2 we begin by describing **partDSA** with observed data and default settings and then describe the user-defined control parameters. Subsequently, in Section 5 we detail examples.

2. partDSA

In the prediction problem, we are interested in building and evaluating the performance of a rule or procedure fitted to n independent observations, corresponding to the n independent subjects in a study. Accordingly, we observe a random sample of n *i.i.d.* observations $W_1, \ldots, W_n$, where $W = (Y, X)$ contains an outcome Y and a collection of p measured explanatory variables, or features, $X = (X_1, ..., X_p)'$. For example, in microarray experiments X includes RNA or protein expression, chromosomal amplification and deletions, or epigenetic changes; while in proteomic data, it includes the intensities at the mass over charge (m/z) values. The collection of features may also contain explanatory variables measured in the clinic and/or by histopathology such as a patient's age or tumor stage. We denote the distri-
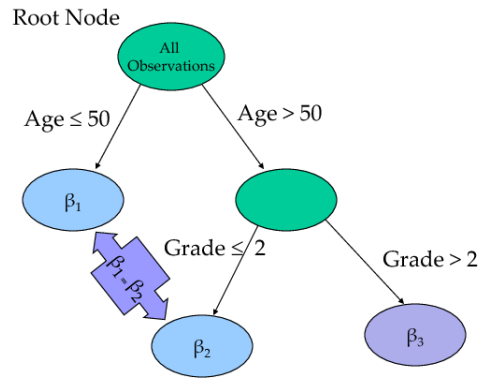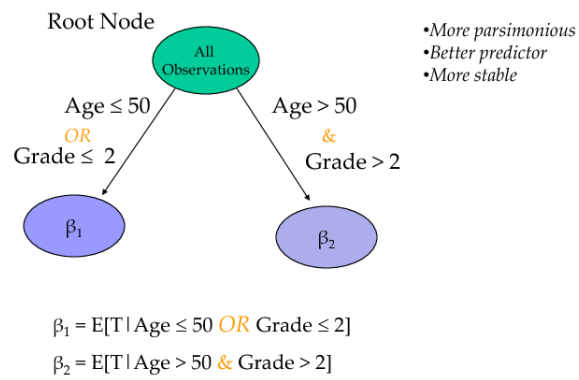
Figure 2: *Two terminal nodes with similar outcome.*

Figure 3: *Example of 'or' statement.*

bution of the data structure W by F_W. The variables which constitute X can be measured on a continuous, ordinal, or categorical scale. Although this covariate process may contain both time-dependent and time-independent covariates we will focus on the time-independent X. The outcome Y may be a continuous measure such as tumor size, a categorical or ordinal measure such as stage of disease, or a binary measure such as disease status.

For an example we will generate a training sample with a categorical outcome Y and two continuous covariates, X_1 and X_2.

```
> y.out=as.factor(sample(c("a", "b", "c"), 50, TRUE) )
> x1=rexp(50)
> x2=runif(50)
```

To load the **partDSA** package and run the algorithm with default settings we type:

```
> library(partDSA)
> #model1<-partDSA(x=data.frame(x1,x2),y=y.out)
```

If an independent test set is available it can be provided to assess how well the model built on the training set performs on the independent test set. For illustration we will generate a larger test set from the same distribution as the training set:

```
> y.out.test=as.factor(sample(c("a", "b", "c"), 100, TRUE) )
> x1.test=rexp(100)
> x2.test=runif(100)
```

The independent test set is included as:

```
> model2<-partDSA(x=data.frame(x1,x2),y=y.out,x.test=data.frame(x1=x1.test,x2=x2.test),y.t
```

If **x.test** and **y.test** are not specified the default values are **x** and **y**, respectively. Case weights can also be specified for the training set via the **wt** argument and for the test set via the **wt.test** argument. By default both are set to vectors of 1's of length equal to the training and test sets, respectively.

2.1. Parallel Computing

Cross-validation is employed in order to select the best model in **partDSA**. To address this added computational burden an optional cluster object can be specified via the sleigh argument which allows the cross-validation to be performed in parallel using the **parallel** package. By default the cross-validation is run sequentially.

An example of running **partDSA** in parallel with two workers is:

```
library(parallel)
cl <- makeCluster(2, type='PSOCK')
cat(sprintf('Running a PSOCK cluster with %d workers\n', length(cl)))
model3<-partDSA(x=data.frame(x1,x2),y=y.out,sleigh=cl)
```

3. Control Parameters

In addition to the training and test sets and corresponding weights, the user can specify numerous parameters via the control object. The control object is created by calling the DSA.control function.

```
DSA.control(vfold=10, minsplit=20, minbuck=round(minsplit/3), cut.off.growth=10, MPD=0.1,
            missing="default", loss.function="default")
```

The default values for each of the parameters are chosen by calling DSA.control with no arguments, i.e.

```
> partDSA(x=data.frame(x1,x2),y=y.out,control=DSA.control())
```

3.1. vfold

Cross-validation is used to select the best model among those generated (Molinaro, Simon, and Pfeiffer 2005; Molinaro and Lostritto 2010). As such, **partDSA** initially splits the training set into v-folds. For each of v times, $v-1$ of the folds are used to build up to $M =cut\text{-}off\text{-}growth$ models (each of which is the best model for $1, \cdots, M$ partitions). Subsequently, the one fold left out of model building is used to assess the performance of the associated model. Once all v-folds have run, the cross-validated error is calculated as the average squared-error (for a continuous outcome) or misclassification error (for a categorical outcome) over the v-folds for each of the possible partitions, $1, \cdots, M$. By default $vfold = 10$ and 10-fold cross validation is performed. For a training set of size n, if $vfold$ is set equal to n it is equivalent to running leave-one-out cross-validation. For exploratory purposes and if no model selection is needed, $vfold$ can be set to 1 to avoid any cross-validation. That is, to just build a model without cross-validation, we can specify:

```
> partDSA(x=data.frame(x1,x2),y=y.out,control=DSA.control(vfold=1))
```

3.2. minsplit

Minsplit is the minimum number of observations necessary to split one partition into two partitions. By default minsplit is set equal to 20, to specify a minimum of 15 observations, we can write:

```
> partDSA(x=data.frame(x1,x2),y=y.out,control=DSA.control(minsplit=15))
```

3.3. minbuck

Minbuck is the minimum number of observations that can be in a partition. In order to split a partition into two (either via the Addition or Substitution steps), there must be at least $2 * minbuck$ observations. By default minbuck is set equal to $round(minsplit/3) = 7$ as $minsplit = 20$ by default. To specify a minimum of 15 observations in any terminal partition, we should also increase minsplit from the default of 20, and we can write:

```
> partDSA(x=data.frame(x1,x2),y=y.out,control=DSA.control(minsplit=40, minbuck=15))
```

3.4. cut-off-growth

Cut-off-growth is the maximum number of partitions that will be explored. The more partitions that are examined the more the computational burden and likelihood of overfitting, thus cut-off-growth has a default value of 10. To specify a different value, we type:

```
> partDSA(x=data.frame(x1,x2),y=y.out,control=DSA.control(cut.off.growth=15))
```

3.5. MPD

Minimum percent difference, or MPD, is the smallest percentage by which the next move (Addition, Substitution, or Deletion) must improve the current fit (i.e., the reduction in the empirical risk). The larger the percentage is the bigger the improvement and the fewer possible moves to consider. By default $MPD = 0.1$, or 10%. To change MPD to 30%, we can specify:

```
> partDSA(x=data.frame(x1,x2),y=y.out,control=DSA.control(MPD=0.3))
```

3.6. Missing values

Currently, **partDSA** can accomodate missing values in the covariates via imputation. As such, there are two settings for the missing value argument named *missing*: "no" and "impute.at.split". *Missing* set to "no" indicates that there is no missing data and will create an error if missing data is found in the dataset. By default *missing = "no"*. For missing covariate data, **partDSA** will employ a data imputation method similar to that in CRUISE (Kim and Loh, 2001) with *missing = "impute.at.split"*. Where at each split, the non-missing observations for a given variable are used to find the best split, and the missing observations are imputed based on the mean or mode (depending on whether the variable is categorical or continuous) of the non-missing observations in that node. Once the node assignment of the missing observations is determined using the imputed values, the imputed values are returned to their missing state. For missing values in the test set, the grand mean or mode from the corresponding variables in the training set are used. Including variables which are entirely missing will result in an error. To set *missing* to "impute.at.split", we type:

```
> partDSA(x=data.frame(x1,x2),y=y.out,control=DSA.control(missing="impute.at.split"))
```

3.7. Loss Functions for Univariate Prediction

partDSA employs loss functions for two key stages of model building: separating the observed sample into different groups (or partitions) and selecting the final prediction model. The general purpose of the loss function L is to quantify performance. Thus, depending on the parameter of interest, there could be numerous loss functions from which to choose. If the outcome Y

is continuous, frequently, the parameter of interest is the conditional mean $\psi_0(W) = E[Y \mid W]$ which has the corresponding squared error loss function, $L(X, \psi) = (Y - \psi(W))^2$.

If the outcome Y is categorical, the parameter of interest involves the class conditional probabilities, $Pr_0(y|W)$. For the indicator loss function, $L(X, \psi) = I(Y \neq \psi(W))$, the optimal parameter is $\psi_0(W) = \text{argmax}_y Pr_0(y \mid W)$, the class with maximum probability given covariates W. One could also use a loss function which incorporates differential misclassification costs. Note that in the standard CART methodology, Breiman *et al.* (1984) favor replacing the indicator loss function in the splitting rule by measures of node impurity, such as the entropy, Gini, or twoing indices (Chapter 4). The indicator loss function is still used for pruning and performance assessment. It turns out that the entropy criterion corresponds to the negative log-likelihood loss function, $L(X, \psi) = -\log \psi(X)$, and parameter of interest $\psi_0(X) = Pr_0(Y|W)$. Likewise, the Gini criterion corresponds to the loss function $L(X, \psi) = 1 - \psi(X)$, with parameter of interest $\psi_0(X) = 1$ if $Y = \text{argmax}_y Pr_0(y \mid W)$ and 0 otherwise. At the current time **partDSA** is enabled for the L_2 or squared error loss function (the default) for continuous outcomes and either the Gini ("gini") or entropy ("entropy", the default) loss functions for categorical outcomes. The loss function can be specified in `DSA.control`:

```
> model2<-partDSA(x=data.frame(x1,x2),y=y.out,
                  control=DSA.control(loss.function="gini"))
```

4. Output and Viewing the partitions

Once **partDSA** has run the outcome can be examined by using the `print()` statement. For example, using the previously generated data we can run the algorithm with only up to two partitions:

```
> model4<-partDSA(x=data.frame(x1,x2),y=y.out,control=DSA.control(missing="no",cut.off.gro
```

And then look at the results by typing:

```
> print(model4)

partDSA object
# partitions    mean CV error    sd CV error    test risk
1               0.616667         0.052705       0.620000
2               0.680000         0.222333       0.560000

Outcome:
[1] a
Levels: a b c
[1] b a
Levels: a b c

Best 2 partitions
```

```
  Partition 1 [of 2]:
     (x1 <= 1.367271)
  Partition 2 [of 2]:
     (1.367271 < x1)


Variable importance matrix:
    COG=1 COG=2
x1      0      2
x2      0      0
```

The first item is the table summarizing the average cross-validated error over the vfolds, the standard error for the cross-validated error, and the test set risk. Each is reported for up to $M = 2 = cut.off.growth$ partitions. The second item labeled 'Outcome' is the predicted outcome for each partition. In our example, if only one partition is chosen (i.e. all observations are together) the predicted outcome is "a". If two partitions are chosen then the first partition has predicted outcome "b" and the second has predicted outcome "a". The description of how the partitions are defined follows. In this example we specified a maximum of 2 partitions, thus, we only see the description for the best of 2 partitions. The final item is the variable importance matrix. This matrix simply keeps track of how many times each variable defines a partition. The variables are listed as rows and the partitions as columns (here COG = cut-off-growth). For example, for COG = 1 neither of the variables define the one partition because all observations are kept in one partition with no splitting. For COG = 2, x1 is used to define 2 of the two total partitions and x2 is used to define 0 of the two total partitions.

In order to view the partitions Java must be installed. For example to see a the two partitions for our running example we type:

```
> showDSA(model4)
```

There are three windows in the output shown in Figure 4(a). The Node Browser, i.e. the top-left window, lists the best of each number of allowed partitions (from $1, \cdots, M = cut - off - growth$) and allows the user to navigate through the different splits and into the final partitions by clicking. The right window displays what is chosen in the Node Browser window. For example, here we see that in the Node Browser window 'Best two partitions' is highlighted and in the right window we see the visual of those two partitions. Namely, observations with a value of x1 greater than 0.77 are assigned to the left partition with predicted outcome "b". Observations with an x1 value less than 0.77 are assigned to the right partition with predicted outcome "a".

The bottom-left window, labeled 'Selected Node', becomes active once a final partition is highlighted in the Node Browser window. This is shown in Figure 4(b). Now note that in the Node Browser window the final partition labeled with the predicted outcome "b" is highlighted and the pathway from the orignating partition to the final partition is highlighted in the right window. The Selected Node window is now active and reports information such as the predicted value and how many observations from the training set are contained in the final partition and thus estimate the predicted value. Additional information such as the

section and number of sections inform the user as to if this final partition stands alone or is part of an 'or' statement ordering.

5. Examples

5.1. Categorical outcome

The German Breast Cancer Study Group (GBSG2) datai is a prospective controlled clinical trial on the treatment of node positive breast cancer patients (Schumacher, Hollander, Schwarzer, and Sauerbrei 2001). The data (available within the TH.data package) contains 686 women with seven prognostic factors measured. For this analysis we will predict recurrence by using the censoring status (0 = uncensored, 1 = censored).

```
> data("GBSG2", package = "TH.data")
> mdl1<-partDSA(x=data.frame(GBSG2[,c(1:8)]),y=as.factor(GBSG2$cens),control=DSA.control(c
> print(mdl1)

partDSA object
# partitions    mean CV error    sd CV error    test risk
1               0.435867         0.002864       0.435860
2               0.396306         0.054824       0.362974
3               0.370241         0.036825       0.336735
4               0.387868         0.044458       0.381924
5               0.392258         0.047392       0.381924

Outcome:
[1] 0
Levels: 0 1
[1] 0 1
Levels: 0 1
[1] 0 0 1
Levels: 0 1
[1] 0 0 0 1
Levels: 0 1
[1] 0 0 0 1 0
Levels: 0 1

Best 2 partitions
  Partition 1 [of 2]:
    (pnodes <= 3.000000)
  Partition 2 [of 2]:
    (3.000000 < pnodes)
Best 3 partitions
  Partition 1 [of 3]:
    (pnodes <= 3.000000)
  Partition 2 [of 3]:
```

```
    (3.000000 < pnodes) && (21.000000 < progrec)
    (tsize <= 20.000000) && (3.000000 < pnodes <= 9.000000) && (progrec <= 21.000000)
  Partition 3 [of 3]:
    (9.000000 < pnodes) && (progrec <= 21.000000)
    (20.000000 < tsize) && (3.000000 < pnodes <= 9.000000) && (progrec <= 21.000000)
Best 4 partitions
  Partition 1 [of 4]:
    (pnodes <= 3.000000) && (progrec <= 89.000000)
  Partition 2 [of 4]:
    (3.000000 < pnodes) && (21.000000 < progrec)
    (tsize <= 20.000000) && (3.000000 < pnodes <= 9.000000) && (progrec <= 21.000000)
  Partition 3 [of 4]:
    (pnodes <= 3.000000) && (89.000000 < progrec)
    (9.000000 < pnodes) && (progrec <= 21.000000) && (estrec <= 0.000000)
    (20.000000 < tsize) && (3.000000 < pnodes <= 9.000000) && (progrec <= 21.000000)
  Partition 4 [of 4]:
    (9.000000 < pnodes) && (progrec <= 21.000000) && (0.000000 < estrec)
Best 5 partitions
  Partition 1 [of 5]:
    (pnodes <= 3.000000) && (progrec <= 89.000000)
  Partition 2 [of 5]:
    (3.000000 < pnodes) && (21.000000 < progrec)
    (tsize <= 20.000000) && (3.000000 < pnodes <= 9.000000) && (progrec <= 21.000000)
  Partition 3 [of 5]:
    (horTh is no) && (pnodes <= 3.000000) && (89.000000 < progrec)
    (9.000000 < pnodes) && (progrec <= 21.000000) && (estrec <= 0.000000)
    (20.000000 < tsize) && (3.000000 < pnodes <= 9.000000) && (progrec <= 21.000000)
  Partition 4 [of 5]:
    (9.000000 < pnodes) && (progrec <= 21.000000) && (0.000000 < estrec)
  Partition 5 [of 5]:
    (horTh is yes) && (pnodes <= 3.000000) && (89.000000 < progrec)


Variable importance matrix:
        COG=1 COG=2 COG=3 COG=4 COG=5
horTh       0     0     0     0     2
age         0     0     0     0     0
menostat    0     0     0     0     0
tsize       0     0     2     2     2
tgrade      0     0     0     0     0
pnodes      0     2     3     4     5
progrec     0     0     2     4     5
estrec      0     0     0     2     2

> showDSA(mdl1)
```

An example of the partDSA model is shown in Figure 5. The following can be read from the visualization:

- If the patient has less than 3 positive nodes (pnodes) she has a predicted outcome of 0 (p1: 0).

- If the patient has greater than 21 positive nodes (pnodes) **OR** between 9 and 21 positive nodes and has estrogen receptor value less than or equal to 0 fmol (estrec) **OR** between 3 and 9 positive nodes and tumor size < 20 and age < 54, she has a predicted outcoem of 0 (p2:0).

- If the patient has between 9 and 21 positive nodes (pnodes) and an estrogen receptor value greater than 0 (estrec), she has a predicted outcome of 1 (p3:1).

- If the patient has between 3 and 9 positive nodes (pnodes) and tumor size greater than 20, she has a predicted outcome of 1 (p4:1).

References

Breiman L, Friedman JH, Olshen R, Stone CJ (1984). *Classification and Regression Trees*. Wadsworth and Brooks/Cole, Monterey, CA.

Molinaro AM, Lostritto K (2010). *Statistical Bioinformatics: A Guide for Life and Biomedical Science Researchers*, chapter Statistical resampling for large screening data analysis such as classical resampling, Bootstrapping, Markov chain Monte Carlo, and statistical simulation and validation strategies. John Wiley & Sons, Inc.

Molinaro AM, Lostritto K, van der Laan MJ (2010). "*partDSA*: Deletion/Substitution/Addition Algorithm for Partitioning the Covariate Space in Prediction." *Bioinformatics*. Doi: 10.1093/bioinformatics/btq142.

Molinaro AM, Lostritto K, Weston S (2009). "partDSA: Partitioning using deletion, substitution, and addition moves." http://cran.r-project.org/web/packages/partDSA.

Molinaro AM, Simon R, Pfeiffer RM (2005). "Prediction error estimation: a comparison of resampling methods." *Bioinformatics*, **21**(15), 3301–3307. http://bioinformatics.oxfordjournals.org/cgi/reprint/21/15/3301.pdf, URL http://bioinformatics.oxfordjournals.org/cgi/content/abstract/21/15/3301.

Schumacher M, Hollander N, Schwarzer G, Sauerbrei W (2001). *Statistics in Oncology*, chapter Prognostic Factor Studies. Marcel Dekker.

Affiliation:

Givenname Familyname
Affiliation
Address, Country
E-mail: name@address
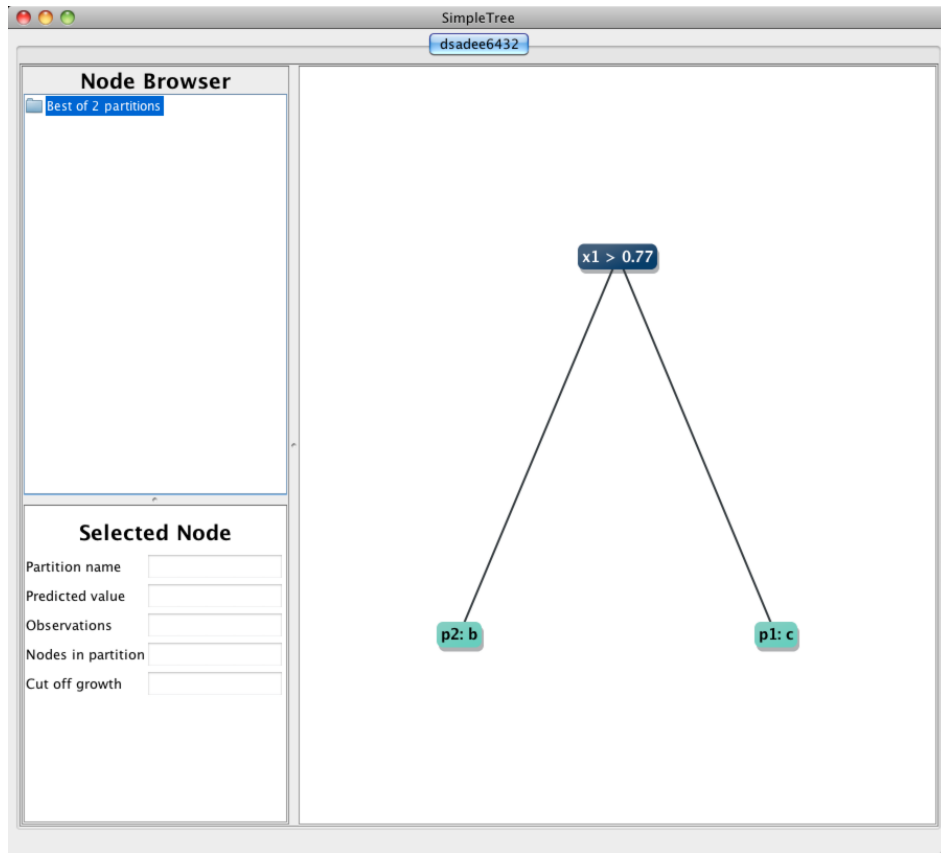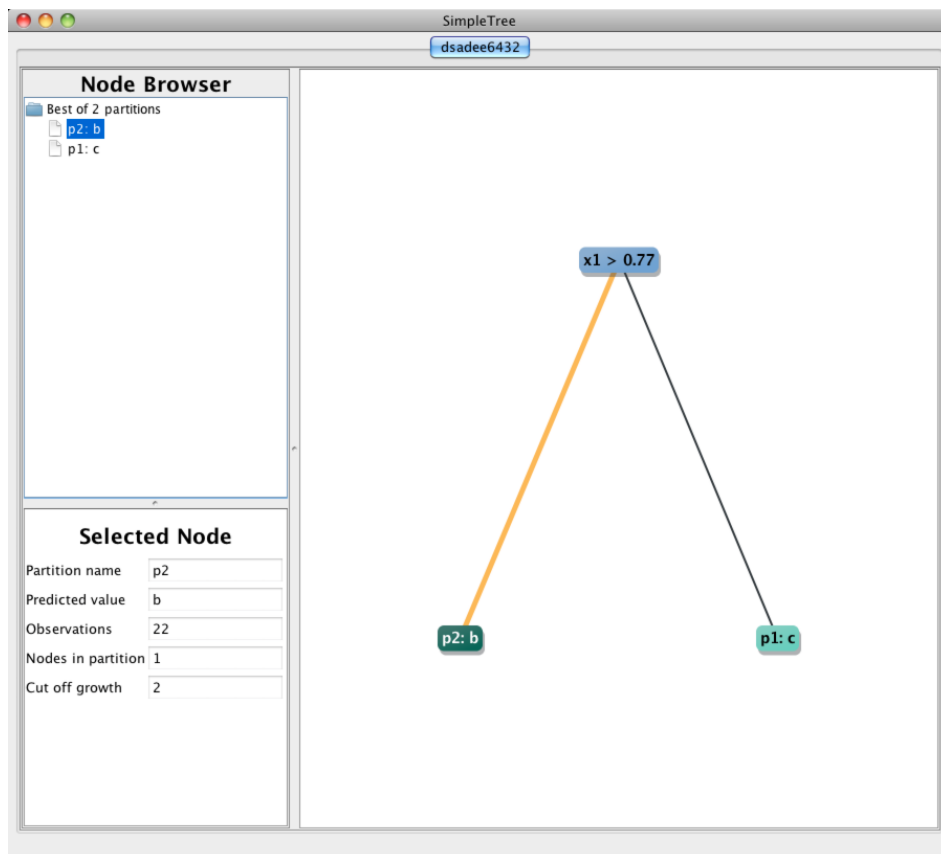URL: https://link/to/webpage/

(a) *Visualization of Model4 with two partitions.*

(b) *Visualization of Model4 with two partitions and one final partition highlighted.*
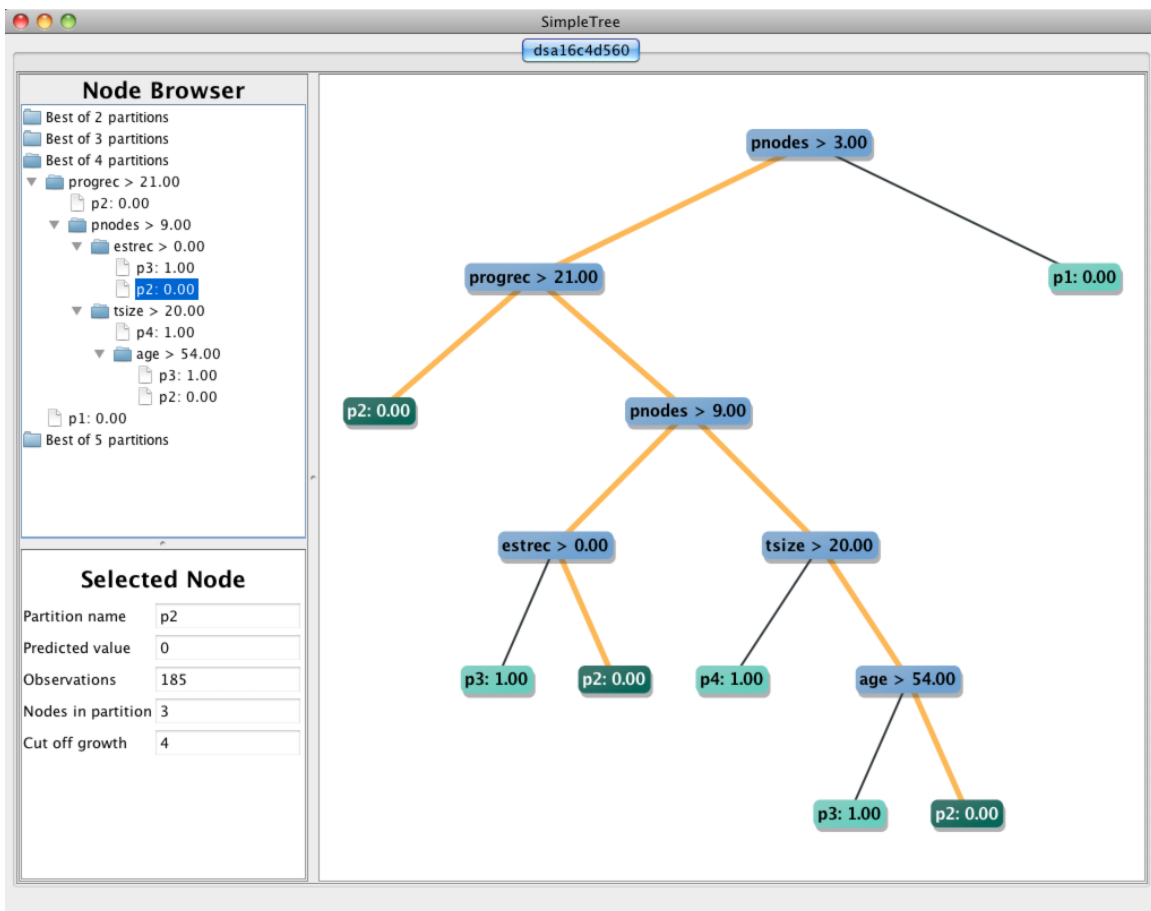
Figure 4: Visualization of Partitions

Figure 5: *German Breast Cancer Study Group Example*