

rpartOrdinal: An R Package for Deriving a Classification Tree for predicting an Ordinal Response

Kellie J. Archer

Virginia Commonwealth University

Abstract

This paper describes an R package, **rpartOrdinal**, that implements alternative splitting functions for fitting a classification tree when interest lies in predicting an ordinal response. This includes the generalized Gini impurity function, which was introduced as a method for predicting an ordinal response by including costs of misclassification into the impurity function, as well as an alternative ordinal impurity function due to [Piccarreta \(2008\)](#) that does not require the assignment of misclassification costs. The ordered twoing splitting method, which is not defined as a decrease in node impurity, is also included in the package. Since, in the ordinal response setting, misclassifying observations to adjacent categories is a less egregious error than misclassifying observations to distant categories, this package also includes a function for estimating an ordinal measure of association, the gamma statistic.

Keywords: machine learning, classification trees, recursive partitioning, ordinal response, R.

1. Introduction

For many high-throughput genomic studies, the phenotype to be fitted is ordinal. Some examples of ordinal responses include the more recently advocated method for evaluating response to treatment in target tumor lesions, known as the response evaluation criteria in solid tumors (RECIST) method, with ordinal outcomes defined as complete response > partial response > stable disease > progressive disease. Moreover, most histopathological measures are ordinal, such as scoring methods for liver biopsy specimens from patients with chronic hepatitis, including the Knodell hepatic activity index, the Ishak score, and the METAVIR score. Statistical methods such as adjacent category, proportional odds, and continuation ratio models ([Agresti 2002](#)) are traditionally used when modeling an ordinal response, though they fail for high-throughput genomic datasets when the number of covariates, p , exceeds the number of observations, n .

An alternative class prediction method, classification trees (CTs), is capable of predicting a response when the $n \ll p$ ([Breiman et al. 1984](#)). Suppose n independent observations to be classified are characterized by a p -dimensional vector of fittedors $\mathbf{x}_i = (x_{i1}, x_{i2}, \dots, x_{ip})$ and each observation \mathbf{x}_i falls into one of J classes. Let ω denote the class with $\omega = \omega_1$ representing observations in class 1, $\omega = \omega_2$ representing class 2, \dots , and $\omega = \omega_J$ representing class J . When deriving a CT, all observations start together in the root node, t . Then, for

fittedors $1, 2, \dots, p$, the optimal split is determined, where optimality is defined as that split resulting in the largest decrease in node impurity.

For node t , the optimal split divides the observations to the left and right descendent nodes, t_L and t_R , respectively, and the proportion of cases in each of the J classes within these nodes are called the node proportions, that is, $p(\omega_j|t)$ for $j = 1, \dots, J$ such that $p(\omega_1|t) + p(\omega_2|t) + \dots + p(\omega_J|t) = 1$. For nominal response classification, the within-node impurity measure most commonly used is the Gini criterion (Breiman *et al.* 1984), defined as

$$i(t) = \sum_k \sum_{k \neq l} p(\omega_k|t)p(\omega_l|t). \quad (1)$$

This is the default impurity function in the R programming environment **rpart** package (R Development Core Team 2009; Therneau and Atkinson 1997) for predicting a nominal class response. However, use of this impurity function does not take advantage of the additional information present when the response is ordinal. To that end, the generalized Gini impurity function (Breiman *et al.* 1984),

$$i_{GG}(t) = \sum_k \sum_{k \neq l} C(\omega_k|\omega_l)p(\omega_k|t)p(\omega_l|t), \quad (2)$$

which factors in $C(\omega_k|\omega_l)$, the cost of misclassifying a class l observation as belonging to class k , has been proposed for ordinal response prediction.

Another proposed ordinal impurity function for deriving an ordinal response classification tree based on a measure of nominal-ordinal association (Piccarreta 2001) that does not require the assignment of costs of misclassification is

$$i_{OS}(t) = \sum_{j=1}^J F(\omega_j|t) (1 - F(\omega_j|t)) \quad (3)$$

where $F(\omega_j|t) = \sum_{k=1}^j p(\omega_k|t)$ (Piccarreta 2008).

A splitting method for ordinal response prediction that is not impurity-based is the ordered twoing method (Breiman *et al.* 1984). Though this method was described in the seminal book by Breiman *et al.* (1984) and has been implemented in the CART Software by Salford Systems (Steinberg and Colla 1997; Steinberg and Golovnya 2006), it has not yet been implemented in R. Ordered twoing proceeds by reformulating the ordinal response as a vector of dichotomous responses, where for each observation i , the j^{th} dichotomous response is taken to be

$$C_{ij} = \begin{cases} 1 & \text{if } \omega_i = 1, \dots, j \\ 0 & \text{if } \omega_i = j + 1, \dots, J. \end{cases} \quad (4)$$

For node t and dichotomous response C_j , the split s that maximizes

$$\phi(s, t, C_j) = 2p_L p_R (p(C_j|t_L) - p(C_j|t_R))^2 \quad (5)$$

over the p covariates is taken to be the best split for that dichotomous response C_j . Subsequently, the split s associated with the dichotomous response

$$j^* = \arg \max_j \phi(s, t, C_j) \quad (6)$$

is then selected for splitting node t . In this paper, we describe the **rpartOrdinal** R package, which implements ordered twoing, the generalized Gini, and the ordinal impurity splitting methods. These splitting methods should be considered for use when deriving an ordinal response classification tree.

For nominal response prediction, misclassification rates are often examined as a means for assessing the performance of the classifier. For ordinal response prediction problems, it may be of more interest to estimate the gamma statistic as an ordinal measure of association between the observed and fitted responses as a means for gauging the success of ordinal classification. Briefly, the association between two ordinal variables X and Y can be estimated by the gamma statistic (Agresti 2002), where given the cross-tabulation matrix T of X and Y having r rows and c columns, the number of concordant pairs for cells $(1, 1)$ to $(r - 1, c - 1)$ is given by

$$C_{kl} = T_{kl} \times \sum_{j=l+1}^c \sum_{i=k+1}^r T_{ij}. \quad (7)$$

Similarly, the number of discordant pairs for cells $(1, 2)$ to $(r - 1, c)$ is given by

$$D_{kl} = T_{kl} \times \sum_{j=1}^{l-1} \sum_{i=k+1}^r T_{ij}. \quad (8)$$

We then let $C = \sum_{j=1}^{c-1} \sum_{i=1}^{r-1} C_{ij}$ and $D = \sum_{j=2}^c \sum_{i=1}^{r-1} D_{ij}$ such that the gamma statistic of ordinal association is defined as

$$\hat{\gamma} = \frac{C - D}{C + D}. \quad (9)$$

An R package, **rpartOrdinal**, that implements the described ordinal splitting methods as well as a function for estimating the gamma statistic as an ordinal measure of association is available for download from the Comprehensive R Archive Network at <http://cran.r-project.org/package=rpartOrdinal>. This package has also been described in Archer (2010)

2. Illustrative dataset

2.1. Low birthweight dataset

The `lowbwt` dataset was downloaded from ftp://ftp.wiley.com/public/sci_tech_med/logistic/ and includes birthweight and associated risk factors measured on 189 women as described by (Hosmer and Lemeshow 2000). For illustrative purposes, an ordinal response variable (`Category`) will be derived from birthweight as defined in Table 1 and added to the `lowbwt` dataset.

In addition to this ordinal response, the dataset includes variables listed in Table 2.

3. Implementation

The **rpartOrdinal** package was written in the R programming environment (R Development Core Team 2009) and depends on the **rpart** package (Therneau and Atkinson 1997). Currently, **rpart** includes methods for deriving regression, classification, and survival trees. Due

1	$\text{bwt} > 3500$
2	$3000 < \text{bwt} \leq 3500$
3	$2500 < \text{bwt} \leq 3000$
4	$\text{bwt} \leq 2500$

Table 1: Ordinal response levels for low birthweight (`Category`).

Variable	Description
<code>low</code>	Dichotomous outcome: Not low birthweight or Low birthweight (<2,500 grams)
<code>age</code>	Age of mother, years
<code>lwt</code>	Mother's weight at last menstrual period, pounds
<code>race</code>	Race of mother (white, black, other)
<code>smoke</code>	Mother's smoking status (No, Yes)
<code>ptl</code>	Number of previous premature labours
<code>ht</code>	Mother's history of hypertension (No, Yes)
<code>ui</code>	Presence of uterine irritability (No, Yes)
<code>ftv</code>	Number of physician visits during the first trimester
<code>bwt</code>	Birth weight in grams

Table 2: Description of covariates included in the low birthweight dataset.

to the `method=` option in `rpart`, users can define their own splitting methods for use in conjunction with the `rpart` function. A user defined method passed to the `method=` option must be a list consisting of three functions named `eval`, `split`, and `init`. Since previous research comparing the ordinal splitting methods to traditional methods for single trees and for bootstrap aggregating classification trees has demonstrated that when the response to be fitted is ordinal, an ordinal splitting method is usually preferred (Piccarreta 2008; Archer and Mas 2009), we have implemented three ordinal splitting methods, namely, ordered twoing, ordinal impurity, and generalized Gini for use in conjunction with `rpart`.

3.1. Ordered twoing

The ordered twoing splitting criteria in Equation 5 has been implemented as a callable method in `rpart`. Here we derive an ordinal classification tree for predicting the ordinal response in the low birthweight dataset, `Category`, using ordered twoing by additionally specifying `method=twoing`. Such a CT may be useful for exploring factors related to poor neonatal outcomes.

```
> library("rpartOrdinal")
> data("lowbwt")
> lowbwt$Category <- factor(ifelse(lowbwt$bwt <= 2500, 3, ifelse(lowbwt$bwt <= 3000, 2, if
> otwoing.rpart <- rpart(Category ~ age + lwt + race + smoke + ptl + ht + ui + ftv, data =
> fitted(otwoing.rpart)
```

```
4 10 11 13 15 16 17 18 19 20 22 23 24 25 26 27 28
3 3 3 3 3 1 3 1 3 3 3 3 1 1 3 0 3
```

```

29 30 31 32 33 34 35 36 37 40 42 43 44 45 46 47 49
 3  3  3  3  3  3  3  0  3  3  3  3  3  0  3  3  1
50 51 52 54 56 57 59 60 61 62 63 65 67 68 69 71 75
 3  3  3  3  2  0  3  3  3  3  1  3  2  0  3  1  3
76 77 78 79 81 82 83 84 85 86 87 88 89 91 92 93 94
 3  3  1  2  3  1  3  3  3  1  0  3  3  1  0  3  2
95 96 97 98 99 100 101 102 103 104 105 106 107 108 109 111 112
 3  3  1  3  3  0  0  3  2  3  2  1  3  0  1  3  0
113 114 115 116 117 118 119 120 121 123 124 125 126 127 128 129 130
 0  0  3  1  1  2  3  0  1  2  0  3  2  2  3  0  1
131 132 133 134 135 136 137 138 139 140 141 142 143 144 145 146 147
 0  3  3  0  1  0  1  3  1  3  2  1  1  3  1  3  1
148 149 150 151 154 155 156 159 160 161 162 163 164 166 167 168 169
 1  1  1  0  1  3  1  1  3  1  3  1  1  1  0  1  0
170 172 173 174 175 176 177 179 180 181 182 183 184 185 186 187 188
 2  3  0  0  0  1  1  1  1  3  0  0  0  0  1  3  3
189 190 191 192 193 195 196 197 199 200 201 202 203 204 205 206 207
 0  0  0  0  0  0  0  3  1  0  1  3  0  0  0  1  0
208 209 210 211 212 213 214 215 216 217 218 219 220 221 222 223 224
 1  2  3  0  1  0  1  0  3  0  1  0  0  0  0  0  0
225 226
 0  0
Levels: 0 1 2 3

```

The predicted class can be extracted using the `fitted` function. The fitted CT can be graphically displayed using the `plot()` and `text()` functions. In plots of the tree topology created using these functions, observations meeting the criterion displayed at a given node proceed to the left descendent node while observations not meeting the criterion displayed at a given node proceed to the right descendent node.

```

> plot(otwoing.rpart)
> text(otwoing.rpart, pretty = TRUE)

```

The additional `pretty = TRUE` argument to the `text` function does not abbreviate factors as alphanumeric characters if they appear in the tree. Alternatively, the `post()` function can be used for producing postscript files containing the tree topology which more extensively labels the splits and fitted class for each node.

```

> post(otwoing.rpart, filename = "TwoingLowbwt.ps", use.n = FALSE, title = "", horizontal

```

3.2. Ordinal impurity function

As with the `twoing` function, the ordinal impurity function in Equation 3 has been implemented as a callable method in `rpart`. That is, within the `rpart` function, the user should specify `method=ordinal` to fit an ordinal response classification tree using Equation 3. Again, the fitted class can be extracted using the `fitted` function. The tree growing procedure can be examined by using the `summary` function. Again, `plot` can be used to graphically display the tree.

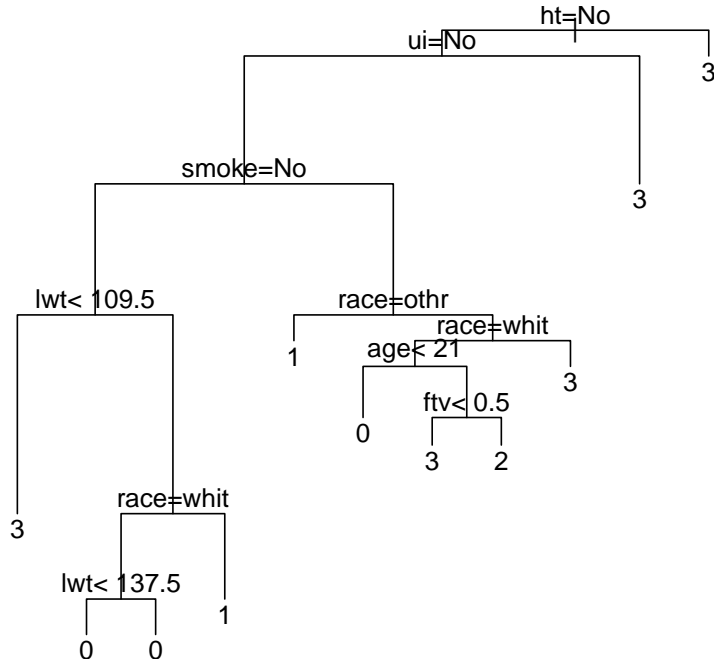


Figure 1: CT for low birthweight dataset using ordered twoing.

```

> ordinal.rpart <- rpart(Category ~ age + lwt + race + smoke + ptl + ht + ui + ftv, data =
> summary(ordinal.rpart)

```

Call:

```

rpart(formula = Category ~ age + lwt + race + smoke + ptl + ht +
      ui + ftv, data = lowbwt, method = ordinal)
n= 189

```

	CP	nsplit	rel error
1	0.04758604	0	1.0000000
2	0.03387924	4	0.8096558
3	0.01406464	5	0.7757766
4	0.01250124	6	0.7617120
5	0.01000000	10	0.7053690

```

Node number 1: 189 observations,      complexity param=0.04758604
predicted class= 3      expected loss= 1.354385

```

left son=2 (177 obs) right son=3 (12 obs)

Primary splits:

ht splits as LR, improve=0.32949840, (0 missing)
 ui splits as LR, improve=0.25190870, (0 missing)
 smoke splits as LR, improve=0.13943480, (0 missing)
 race splits as LRR, improve=0.10658910, (0 missing)
 lwt < 109.5 to the left, improve=0.03974253, (0 missing)

Surrogate splits:

lwt < 232 to the left, agree=0.942, adj=0.083, (0 split)

Node number 2: 177 observations, complexity param=0.04758604

predicted class= 3 expected loss= 1.320566

left son=4 (149 obs) right son=5 (28 obs)

Primary splits:

ui splits as LR, improve=0.23484780, (0 missing)
 smoke splits as LR, improve=0.13239160, (0 missing)
 race splits as LRR, improve=0.10835750, (0 missing)
 lwt < 109.5 to the left, improve=0.04227940, (0 missing)
 ptl < 0.5 to the left, improve=0.02375545, (0 missing)

Surrogate splits:

lwt < 91.5 to the right, agree=0.853, adj=0.071, (0 split)

Node number 3: 12 observations

predicted class= 3 expected loss= 1.666667

Node number 4: 149 observations, complexity param=0.04758604

predicted class= 0 expected loss= 1.306338

left son=8 (93 obs) right son=9 (56 obs)

Primary splits:

smoke splits as LR, improve=0.132017000, (0 missing)
 race splits as LRR, improve=0.120637600, (0 missing)
 lwt < 109.5 to the left, improve=0.058949790, (0 missing)
 ptl < 0.5 to the left, improve=0.027140240, (0 missing)
 age < 27.5 to the left, improve=0.009939474, (0 missing)

Surrogate splits:

ptl < 0.5 to the left, agree=0.658, adj=0.089, (0 split)

lwt < 94.5 to the right, agree=0.644, adj=0.054, (0 split)

Node number 5: 28 observations

predicted class= 3 expected loss= 0.932398

Node number 8: 93 observations, complexity param=0.04758604

predicted class= 0 expected loss= 1.259336

left son=16 (15 obs) right son=17 (78 obs)

Primary splits:

lwt < 109.5 to the left, improve=0.077495840, (0 missing)
 race splits as LRR, improve=0.060046470, (0 missing)

```

age < 27.5 to the left, improve=0.020809060, (0 missing)
ptl < 0.5  to the left, improve=0.015460280, (0 missing)
ftv < 0.5  to the left, improve=0.006709946, (0 missing)

```

```

Node number 9: 56 observations,    complexity param=0.01250124
predicted class= 3    expected loss= 1.146684
left son=18 (8 obs) right son=19 (48 obs)

```

Primary splits:

```

race splits as RRL,          improve=0.17777050, (0 missing)
lwt < 111  to the left, improve=0.05258450, (0 missing)
age < 19.5 to the left, improve=0.02917911, (0 missing)
ptl < 0.5  to the left, improve=0.02508503, (0 missing)
ftv < 1.5  to the left, improve=0.01024764, (0 missing)

```

```

Node number 16: 15 observations
predicted class= 3    expected loss= 1.128889

```

```

Node number 17: 78 observations,    complexity param=0.03387924
predicted class= 0    expected loss= 1.025641
left son=34 (38 obs) right son=35 (40 obs)

```

Primary splits:

```

race splits as LRR,          improve=0.069252460, (0 missing)
age < 19.5 to the left, improve=0.022824900, (0 missing)
ftv < 0.5  to the left, improve=0.011149340, (0 missing)
lwt < 156.5 to the left, improve=0.009577033, (0 missing)

```

Surrogate splits:

```

age < 20.5 to the right, agree=0.654, adj=0.289, (0 split)
lwt < 128.5 to the right, agree=0.654, adj=0.289, (0 split)
ftv < 0.5  to the right, agree=0.628, adj=0.237, (0 split)

```

```

Node number 18: 8 observations
predicted class= 1    expected loss= 0.984375

```

```

Node number 19: 48 observations,    complexity param=0.01250124
predicted class= 3    expected loss= 1.134983
left son=38 (39 obs) right son=39 (9 obs)

```

Primary splits:

```

race splits as LR-,          improve=0.194258400, (0 missing)
lwt < 111  to the left, improve=0.039584060, (0 missing)
age < 22.5 to the left, improve=0.036090730, (0 missing)
ptl < 0.5  to the left, improve=0.023597400, (0 missing)
ftv < 1.5  to the left, improve=0.007811546, (0 missing)

```

```

Node number 34: 38 observations,    complexity param=0.01406464
predicted class= 0    expected loss= 0.8040166
left son=68 (21 obs) right son=69 (17 obs)

```

Primary splits:

lwt < 137.5 to the left, improve=0.051159240, (0 missing)
ftv < 0.5 to the left, improve=0.013622420, (0 missing)
age < 29.5 to the left, improve=0.008755441, (0 missing)

Surrogate splits:

age < 24.5 to the left, agree=0.632, adj=0.176, (0 split)
ftv < 0.5 to the right, agree=0.632, adj=0.176, (0 split)

Node number 35: 40 observations

predicted class= 1 expected loss= 1.019375

Node number 38: 39 observations, complexity param=0.01250124

predicted class= 3 expected loss= 1.188692

left son=76 (16 obs) right son=77 (23 obs)

Primary splits:

lwt < 119 to the left, improve=0.05041377, (0 missing)
age < 21 to the left, improve=0.04875082, (0 missing)
ptl < 0.5 to the left, improve=0.03333174, (0 missing)
ftv < 0.5 to the left, improve=0.00721643, (0 missing)

Surrogate splits:

ptl < 0.5 to the right, agree=0.641, adj=0.125, (0 split)
age < 22.5 to the right, agree=0.615, adj=0.063, (0 split)

Node number 39: 9 observations

predicted class= 3 expected loss= 0.6666667

Node number 68: 21 observations

predicted class= 0 expected loss= 0.61678

Node number 69: 17 observations

predicted class= 0 expected loss= 0.8235294

Node number 76: 16 observations

predicted class= 3 expected loss= 0.5625

Node number 77: 23 observations, complexity param=0.01250124

predicted class= 0 expected loss= 1.379962

left son=154 (11 obs) right son=155 (12 obs)

Primary splits:

age < 21 to the left, improve=0.08192988, (0 missing)
lwt < 141 to the left, improve=0.03341210, (0 missing)
ftv < 0.5 to the left, improve=0.02593515, (0 missing)

Surrogate splits:

lwt < 122.5 to the left, agree=0.652, adj=0.273, (0 split)
ptl < 0.5 to the left, agree=0.609, adj=0.182, (0 split)
ftv < 0.5 to the left, agree=0.609, adj=0.182, (0 split)

Node number 154: 11 observations

```

predicted class= 0   expected loss= 1.454545

Node number 155: 12 observations
predicted class= 2   expected loss= 0.9097222

> fitted(ordinal.rpart)

  4 10 11 13 15 16 17 18 19 20 22 23 24 25 26 27 28
  3  3  3  3  3  1  3  1  3  3  3  3  1  1  3  0  3
29 30 31 32 33 34 35 36 37 40 42 43 44 45 46 47 49
  2  3  3  3  3  3  3  0  3  3  3  3  3  3  3  3  1
50 51 52 54 56 57 59 60 61 62 63 65 67 68 69 71 75
  3  3  3  3  3  0  3  3  3  3  1  2  2  0  3  1  3
76 77 78 79 81 82 83 84 85 86 87 88 89 91 92 93 94
  3  2  1  3  3  1  3  3  3  1  3  3  3  1  0  3  2
95 96 97 98 99 100 101 102 103 104 105 106 107 108 109 111 112
  3  3  1  3  3  3  3  3  3  3  2  1  3  0  1  3  0
113 114 115 116 117 118 119 120 121 123 124 125 126 127 128 129 130
  0  0  3  1  1  3  3  0  1  2  0  2  2  3  3  0  1
131 132 133 134 135 136 137 138 139 140 141 142 143 144 145 146 147
  0  3  3  0  1  0  1  3  1  2  3  1  1  3  1  3  1
148 149 150 151 154 155 156 159 160 161 162 163 164 166 167 168 169
  1  1  1  0  1  3  1  1  3  1  3  1  1  1  0  1  0
170 172 173 174 175 176 177 179 180 181 182 183 184 185 186 187 188
  2  3  0  0  0  1  1  1  1  3  0  0  0  0  1  3  3
189 190 191 192 193 195 196 197 199 200 201 202 203 204 205 206 207
  0  0  0  0  0  0  0  3  1  0  1  3  0  0  0  1  0
208 209 210 211 212 213 214 215 216 217 218 219 220 221 222 223 224
  1  2  3  0  1  0  1  0  3  0  1  0  0  0  0  0  0
225 226
  0  0
Levels: 0 1 2 3

> plot(ordinal.rpart)
> text(ordinal.rpart, pretty = TRUE)

```

The ordered twining and ordinal tree topologies are very similar with exception that node 38 is split by `lwt` in the ordinal tree whereas this same node is split by `age` in the ordered twining tree, with the descendent node 77 splitting variable also differing.

3.3. Generalized Gini impurity

The generalized Gini impurity function in Equation 2 has been implemented in this package by allowing the user to specify a `loss.matrix` parameter in the optional `parms` argument within the `rpart` call. The `loss.matrix` parameter accepts either "linear" or "quadratic" for using either linear or quadratic loss, respectively. The specific syntax for the low birthweight example using the linear loss follows. Note that for the generalized Gini impurity function,

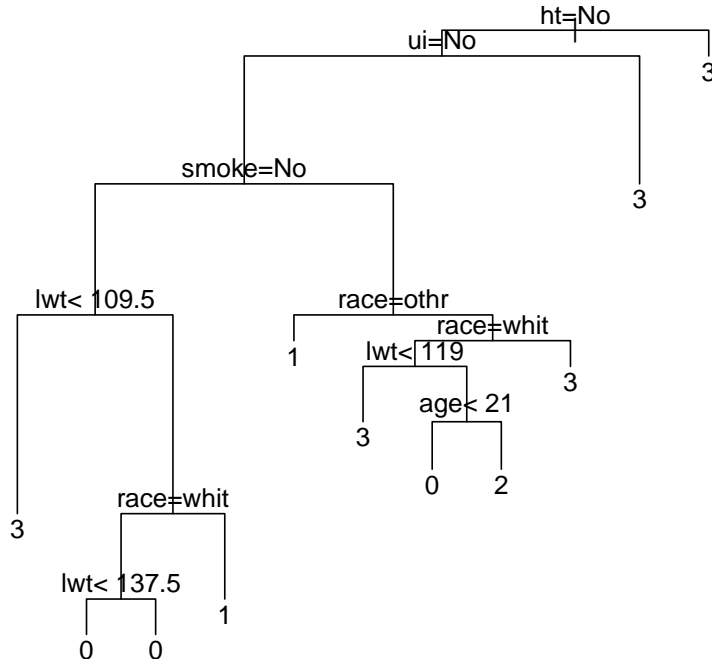


Figure 2: CT for low birthweight dataset using the ordinal impurity function.

`fitted` returns the class probabilities by default. To get the predicted class, the `type="class"` can be used.

```
> linear.loss.rpart <- rpart(Category ~ age + lwt + race + smoke + ptl + ht + ui + ftv, d
> fitted(linear.loss.rpart)
```

	0	1	2	3
4	0.05882353	0.17647059	0.17647059	0.58823529
10	0.05882353	0.17647059	0.17647059	0.58823529
11	0.18965517	0.37931034	0.17241379	0.25862069
13	0.06250000	0.12500000	0.21875000	0.59375000
15	0.06250000	0.12500000	0.21875000	0.59375000
16	0.18965517	0.37931034	0.17241379	0.25862069
17	0.06250000	0.12500000	0.21875000	0.59375000
18	0.18965517	0.37931034	0.17241379	0.25862069
19	0.18965517	0.37931034	0.17241379	0.25862069
20	0.05263158	0.21052632	0.31578947	0.42105263
22	0.06250000	0.12500000	0.21875000	0.59375000

23	0.06250000	0.12500000	0.21875000	0.59375000
24	0.18965517	0.37931034	0.17241379	0.25862069
25	0.18965517	0.37931034	0.17241379	0.25862069
26	0.06250000	0.12500000	0.21875000	0.59375000
27	0.57142857	0.07142857	0.14285714	0.21428571
28	0.05882353	0.17647059	0.17647059	0.58823529
29	0.05263158	0.21052632	0.31578947	0.42105263
30	0.06250000	0.12500000	0.21875000	0.59375000
31	0.05882353	0.17647059	0.17647059	0.58823529
32	0.06250000	0.12500000	0.21875000	0.59375000
33	0.06250000	0.12500000	0.21875000	0.59375000
34	0.05882353	0.17647059	0.17647059	0.58823529
35	0.05263158	0.21052632	0.31578947	0.42105263
36	0.35294118	0.35294118	0.23529412	0.05882353
37	0.05882353	0.17647059	0.17647059	0.58823529
40	0.18965517	0.37931034	0.17241379	0.25862069
42	0.05882353	0.17647059	0.17647059	0.58823529
43	0.05882353	0.17647059	0.17647059	0.58823529
44	0.06250000	0.12500000	0.21875000	0.59375000
45	0.57142857	0.07142857	0.14285714	0.21428571
46	0.06250000	0.12500000	0.21875000	0.59375000
47	0.06250000	0.12500000	0.21875000	0.59375000
49	0.18965517	0.37931034	0.17241379	0.25862069
50	0.18965517	0.37931034	0.17241379	0.25862069
51	0.05882353	0.17647059	0.17647059	0.58823529
52	0.06250000	0.12500000	0.21875000	0.59375000
54	0.06250000	0.12500000	0.21875000	0.59375000
56	0.06250000	0.12500000	0.21875000	0.59375000
57	0.72727273	0.18181818	0.04545455	0.04545455
59	0.18965517	0.37931034	0.17241379	0.25862069
60	0.18965517	0.37931034	0.17241379	0.25862069
61	0.06250000	0.12500000	0.21875000	0.59375000
62	0.05882353	0.17647059	0.17647059	0.58823529
63	0.18965517	0.37931034	0.17241379	0.25862069
65	0.05263158	0.21052632	0.31578947	0.42105263
67	0.05263158	0.21052632	0.31578947	0.42105263
68	0.57142857	0.07142857	0.14285714	0.21428571
69	0.05263158	0.21052632	0.31578947	0.42105263
71	0.18965517	0.37931034	0.17241379	0.25862069
75	0.18965517	0.37931034	0.17241379	0.25862069
76	0.06250000	0.12500000	0.21875000	0.59375000
77	0.05263158	0.21052632	0.31578947	0.42105263
78	0.10000000	0.20000000	0.50000000	0.20000000
79	0.06250000	0.12500000	0.21875000	0.59375000
81	0.10000000	0.20000000	0.50000000	0.20000000
82	0.06250000	0.12500000	0.21875000	0.59375000
83	0.18965517	0.37931034	0.17241379	0.25862069

84 0.05263158 0.21052632 0.31578947 0.42105263
85 0.05882353 0.17647059 0.17647059 0.58823529
86 0.18965517 0.37931034 0.17241379 0.25862069
87 0.06250000 0.12500000 0.21875000 0.59375000
88 0.06250000 0.12500000 0.21875000 0.59375000
89 0.10000000 0.20000000 0.50000000 0.20000000
91 0.18965517 0.37931034 0.17241379 0.25862069
92 0.72727273 0.18181818 0.04545455 0.04545455
93 0.10000000 0.20000000 0.50000000 0.20000000
94 0.05263158 0.21052632 0.31578947 0.42105263
95 0.05263158 0.21052632 0.31578947 0.42105263
96 0.06250000 0.12500000 0.21875000 0.59375000
97 0.18965517 0.37931034 0.17241379 0.25862069
98 0.06250000 0.12500000 0.21875000 0.59375000
99 0.06250000 0.12500000 0.21875000 0.59375000
100 0.10000000 0.20000000 0.50000000 0.20000000
101 0.10000000 0.20000000 0.50000000 0.20000000
102 0.10000000 0.20000000 0.50000000 0.20000000
103 0.05263158 0.21052632 0.31578947 0.42105263
104 0.05882353 0.17647059 0.17647059 0.58823529
105 0.05263158 0.21052632 0.31578947 0.42105263
106 0.18965517 0.37931034 0.17241379 0.25862069
107 0.06250000 0.12500000 0.21875000 0.59375000
108 0.35294118 0.35294118 0.23529412 0.05882353
109 0.18965517 0.37931034 0.17241379 0.25862069
111 0.05882353 0.17647059 0.17647059 0.58823529
112 0.35294118 0.35294118 0.23529412 0.05882353
113 0.57142857 0.07142857 0.14285714 0.21428571
114 0.35294118 0.35294118 0.23529412 0.05882353
115 0.18965517 0.37931034 0.17241379 0.25862069
116 0.18965517 0.37931034 0.17241379 0.25862069
117 0.18965517 0.37931034 0.17241379 0.25862069
118 0.06250000 0.12500000 0.21875000 0.59375000
119 0.18965517 0.37931034 0.17241379 0.25862069
120 0.35294118 0.35294118 0.23529412 0.05882353
121 0.18965517 0.37931034 0.17241379 0.25862069
123 0.05263158 0.21052632 0.31578947 0.42105263
124 0.57142857 0.07142857 0.14285714 0.21428571
125 0.05263158 0.21052632 0.31578947 0.42105263
126 0.05263158 0.21052632 0.31578947 0.42105263
127 0.06250000 0.12500000 0.21875000 0.59375000
128 0.18965517 0.37931034 0.17241379 0.25862069
129 0.35294118 0.35294118 0.23529412 0.05882353
130 0.18965517 0.37931034 0.17241379 0.25862069
131 0.35294118 0.35294118 0.23529412 0.05882353
132 0.10000000 0.20000000 0.50000000 0.20000000
133 0.10000000 0.20000000 0.50000000 0.20000000

```
134 0.72727273 0.18181818 0.04545455 0.04545455
135 0.18965517 0.37931034 0.17241379 0.25862069
136 0.72727273 0.18181818 0.04545455 0.04545455
137 0.06250000 0.12500000 0.21875000 0.59375000
138 0.72727273 0.18181818 0.04545455 0.04545455
139 0.18965517 0.37931034 0.17241379 0.25862069
140 0.05263158 0.21052632 0.31578947 0.42105263
141 0.06250000 0.12500000 0.21875000 0.59375000
142 0.18965517 0.37931034 0.17241379 0.25862069
143 0.18965517 0.37931034 0.17241379 0.25862069
144 0.05882353 0.17647059 0.17647059 0.58823529
145 0.18965517 0.37931034 0.17241379 0.25862069
146 0.06250000 0.12500000 0.21875000 0.59375000
147 0.18965517 0.37931034 0.17241379 0.25862069
148 0.18965517 0.37931034 0.17241379 0.25862069
149 0.18965517 0.37931034 0.17241379 0.25862069
150 0.18965517 0.37931034 0.17241379 0.25862069
151 0.35294118 0.35294118 0.23529412 0.05882353
154 0.18965517 0.37931034 0.17241379 0.25862069
155 0.05882353 0.17647059 0.17647059 0.58823529
156 0.18965517 0.37931034 0.17241379 0.25862069
159 0.18965517 0.37931034 0.17241379 0.25862069
160 0.05882353 0.17647059 0.17647059 0.58823529
161 0.18965517 0.37931034 0.17241379 0.25862069
162 0.05263158 0.21052632 0.31578947 0.42105263
163 0.18965517 0.37931034 0.17241379 0.25862069
164 0.18965517 0.37931034 0.17241379 0.25862069
166 0.18965517 0.37931034 0.17241379 0.25862069
167 0.57142857 0.07142857 0.14285714 0.21428571
168 0.18965517 0.37931034 0.17241379 0.25862069
169 0.35294118 0.35294118 0.23529412 0.05882353
170 0.05263158 0.21052632 0.31578947 0.42105263
172 0.18965517 0.37931034 0.17241379 0.25862069
173 0.35294118 0.35294118 0.23529412 0.05882353
174 0.72727273 0.18181818 0.04545455 0.04545455
175 0.35294118 0.35294118 0.23529412 0.05882353
176 0.18965517 0.37931034 0.17241379 0.25862069
177 0.18965517 0.37931034 0.17241379 0.25862069
179 0.18965517 0.37931034 0.17241379 0.25862069
180 0.18965517 0.37931034 0.17241379 0.25862069
181 0.06250000 0.12500000 0.21875000 0.59375000
182 0.72727273 0.18181818 0.04545455 0.04545455
183 0.35294118 0.35294118 0.23529412 0.05882353
184 0.72727273 0.18181818 0.04545455 0.04545455
185 0.72727273 0.18181818 0.04545455 0.04545455
186 0.18965517 0.37931034 0.17241379 0.25862069
187 0.57142857 0.07142857 0.14285714 0.21428571
```

```

188 0.06250000 0.12500000 0.21875000 0.59375000
189 0.57142857 0.07142857 0.14285714 0.21428571
190 0.72727273 0.18181818 0.04545455 0.04545455
191 0.35294118 0.35294118 0.23529412 0.05882353
192 0.57142857 0.07142857 0.14285714 0.21428571
193 0.57142857 0.07142857 0.14285714 0.21428571
195 0.72727273 0.18181818 0.04545455 0.04545455
196 0.72727273 0.18181818 0.04545455 0.04545455
197 0.57142857 0.07142857 0.14285714 0.21428571
199 0.18965517 0.37931034 0.17241379 0.25862069
200 0.72727273 0.18181818 0.04545455 0.04545455
201 0.18965517 0.37931034 0.17241379 0.25862069
202 0.18965517 0.37931034 0.17241379 0.25862069
203 0.72727273 0.18181818 0.04545455 0.04545455
204 0.35294118 0.35294118 0.23529412 0.05882353
205 0.57142857 0.07142857 0.14285714 0.21428571
206 0.18965517 0.37931034 0.17241379 0.25862069
207 0.35294118 0.35294118 0.23529412 0.05882353
208 0.18965517 0.37931034 0.17241379 0.25862069
209 0.05263158 0.21052632 0.31578947 0.42105263
210 0.05882353 0.17647059 0.17647059 0.58823529
211 0.57142857 0.07142857 0.14285714 0.21428571
212 0.18965517 0.37931034 0.17241379 0.25862069
213 0.72727273 0.18181818 0.04545455 0.04545455
214 0.18965517 0.37931034 0.17241379 0.25862069
215 0.72727273 0.18181818 0.04545455 0.04545455
216 0.10000000 0.20000000 0.50000000 0.20000000
217 0.35294118 0.35294118 0.23529412 0.05882353
218 0.18965517 0.37931034 0.17241379 0.25862069
219 0.72727273 0.18181818 0.04545455 0.04545455
220 0.72727273 0.18181818 0.04545455 0.04545455
221 0.72727273 0.18181818 0.04545455 0.04545455
222 0.72727273 0.18181818 0.04545455 0.04545455
223 0.35294118 0.35294118 0.23529412 0.05882353
224 0.57142857 0.07142857 0.14285714 0.21428571
225 0.72727273 0.18181818 0.04545455 0.04545455
226 0.72727273 0.18181818 0.04545455 0.04545455

```

```
> fitted(linear.loss.rpart,type="class")
```

```

 4  10  11  13  15  16  17  18  19  20  22  23  24  25  26  27  28
 3   3   1   3   3   1   3   1   1   2   3   3   1   1   3   0   3
29  30  31  32  33  34  35  36  37  40  42  43  44  45  46  47  49
 2   3   3   3   3   3   2   1   3   1   3   3   3   0   3   3   1
50  51  52  54  56  57  59  60  61  62  63  65  67  68  69  71  75
 1   3   3   3   3   0   1   1   3   3   1   2   2   0   2   1   1
76  77  78  79  81  82  83  84  85  86  87  88  89  91  92  93  94

```

```

  3  2  2  3  2  3  1  2  3  1  3  3  2  1  0  2  2
95 96 97 98 99 100 101 102 103 104 105 106 107 108 109 111 112
  2  3  1  3  3  2  2  2  2  3  2  1  3  1  1  3  1
113 114 115 116 117 118 119 120 121 123 124 125 126 127 128 129 130
  0  1  1  1  1  3  1  1  1  2  0  2  2  3  1  1  1
131 132 133 134 135 136 137 138 139 140 141 142 143 144 145 146 147
  1  2  2  0  1  0  3  0  1  2  3  1  1  3  1  3  1
148 149 150 151 154 155 156 159 160 161 162 163 164 166 167 168 169
  1  1  1  1  1  3  1  1  3  1  2  1  1  1  0  1  1
170 172 173 174 175 176 177 179 180 181 182 183 184 185 186 187 188
  2  1  1  0  1  1  1  1  1  3  0  1  0  0  1  0  3
189 190 191 192 193 195 196 197 199 200 201 202 203 204 205 206 207
  0  0  1  0  0  0  0  0  0  1  0  1  1  0  1  0  1  1
208 209 210 211 212 213 214 215 216 217 218 219 220 221 222 223 224
  1  2  3  0  1  0  1  0  2  1  1  0  0  0  0  1  0
225 226
  0  0
Levels: 0 1 2 3

```

```

> plot(linear.loss.rpart)
> text(linear.loss.rpart, pretty = TRUE)

```

The specific syntax for the low birthweight example using the quadratic loss function is

```

> quad.loss.rpart <- rpart(Category ~ age + lwt + race + smoke +
+   ptl + ht + ui + ftv, data = lowbwt, method = "class",
+   parms = list(loss = loss.matrix(method = "quad", lowbwt$Category)))
> plot(quad.loss.rpart)
> text(quad.loss.rpart, pretty = TRUE)

```

Both CTs derived using the generalized Gini criteria split the root node using $lwt \geq 109.5$ and split node 2 using $ui=0$. However, other splits differed between the linear and quadratic loss functions.

3.4. Gamma statistic

The `ordinal.gamma` function estimates the gamma statistic, which is a measure of the strength of the association of the cross-tabulation of two ordinal variables. The following example replicates Table 2.8 in Agresti (2000).

```

> library("rpartOrdinal")
> job.satis <- factor(c(1,rep(2,3), rep(3,10), rep(4,6), rep(1,2), rep(2,3), rep(3,10), re
> income <- factor(c(rep(1,20), rep(2,22), rep(3,33), rep(4,21)), ordered = TRUE, labels =
> table(job.satis,income)

```

	income			
job.satis	<15,000	15,000–25,000	25,000–40,000	>40,000

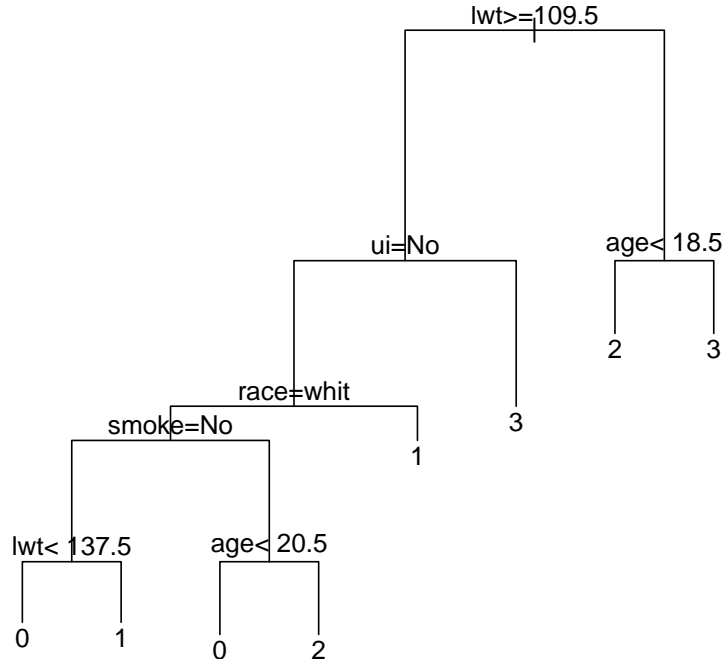


Figure 3: CT for lowbwt using generalized Gini with linear cost of misclassification.

Very Dissatisfied	1	2	1	0
Little Dissatisfied	3	3	6	1
Moderately Satisfied	10	10	14	9
Satisfied	6	7	12	11

```
> ordinal.gamma(job.satis, income)
```

```
[1] 0.2211009
```

Returning to the ordinal classification trees for the low birthweight dataset, it may be of interest to estimate the gamma statistic as an ordinal measure of association between the observed and fitted ordinal responses. However, estimating the gamma statistic as an ordinal measure of association for the training data will not provide useful information regarding how well the fitted model may generalize when presented with new data. Therefore, cross-validation methods may be used. The following code was used to perform five-fold cross-validation where the observations included in the V^{th} fold are stored in the V^{th} component of `groups`. Letting \mathcal{L} represent the full dataset, each method (ordinal, ordered twoing, generalized Gini

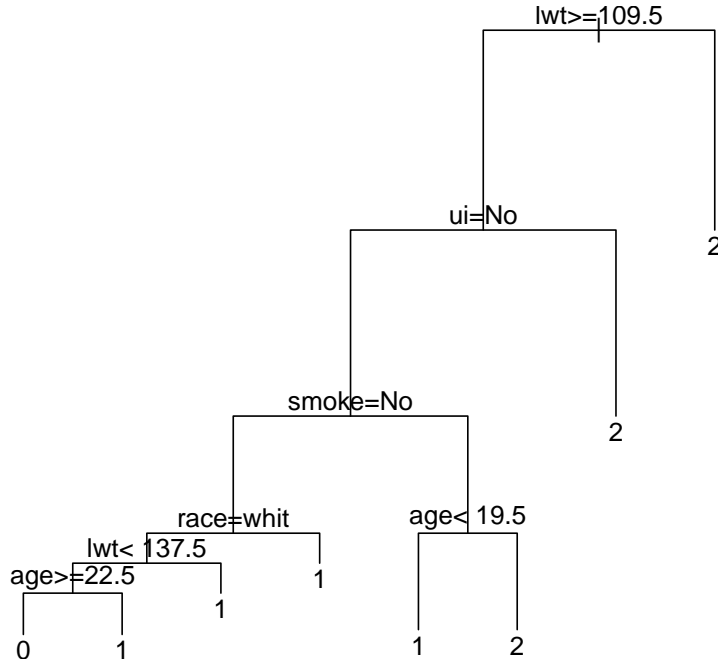


Figure 4: CT for lowbwt using generalized Gini with quadratic cost of misclassification.

with linear loss, and generalized Gini with quadratic loss) was fit using the observations in $\mathcal{L} \setminus \mathcal{L}_v$ then the fitted class was obtained for the observations in \mathcal{L}_v .

```

> V <- 5
> n <- length(lowbwt$Category)
> leave.out <- trunc(n/V)
> o <- sample(1:n)
> groups <- vector("list", V)
> for (j in 1:(V - 1)) {
+   jj <- (1 + (j - 1) * leave.out)
+   groups[[j]] <- (o[jj:(jj + leave.out - 1)])
+ }
> groups[[V]] <- o[(1 + (V - 1) * leave.out):n]
> linear.fit <- rep(NA, n)
> quad.fit <- rep(NA, n)
> ordinal.fit <- rep(NA, n)
> twoing.fit <- rep(NA, n)
> for (j in 1:V) {

```

```

+ ordinal.rpart <- rpart(Category ~ age + lwt + race + smoke +
+   ptl + ht + ui + ftv, data = lowbwt, subset = -groups[[j]],
+   method = ordinal)
+ ordinal.fit[groups[[j]]]<-fitted(ordinal.rpart, newdata = lowbwt[groups[[j]],])
+ twoing.rpart <- rpart(Category ~ age + lwt + race + smoke +
+   ptl + ht + ui + ftv, data = lowbwt, subset = -groups[[j]],
+   method = twoing)
+ twoing.fit[groups[[j]]]<-fitted(twoing.rpart, newdata = lowbwt[groups[[j]],])
+ linear.rpart <- rpart(Category ~ age + lwt + race + smoke +
+   ptl + ht + ui + ftv, data = lowbwt, subset= -groups[[j]],
+   parms = list(loss = loss.matrix(method = "linear", lowbwt$Category)))
+ phat <- fitted(linear.rpart, newdata=lowbwt[groups[[j]],])
+ linear.fit[groups[[j]]] <- apply(phat, 1, which.max)
+ quadratic.rpart <- rpart(Category ~ age + lwt + race + smoke +
+   ptl + ht + ui + ftv, data = lowbwt, subset = -groups[[j]],
+   parms = list(loss = loss.matrix(method = "quad", lowbwt$Category)))
+ phat <- fitted(quadratic.rpart, newdata=lowbwt[groups[[j]],])
+ quad.fit[groups[[j]]] <- apply(phat, 1, which.max)
+ }
> ordinal.gamma(lowbwt$Category, twoing.fit)

[1] 0.4050294

> ordinal.gamma(lowbwt$Category, ordinal.fit)

[1] 0.3963154

> ordinal.gamma(lowbwt$Category, linear.fit)

[1] 0.3015971

> ordinal.gamma(lowbwt$Category, quad.fit)

[1] 0.3608686

```

For this random partition of the `lowbwt` dataset, the ordinal and ordered twoing methods had the similar performance and both performed better than the generalized Gini with either quadratic or linear loss. If the sample size is large, a split sample approach could be used wherein the `rpart` function would be applied to a `train` dataset and the `fitted` function applied using `newdata=test`. Alternatively, one can easily construct a bootstrap procedure and estimate error using out-of-bag observations as in [Archer and Mas \(2009\)](#).

Summary

Herein we have described the **rpartOrdinal** package which works in conjunction with the **rpart** package in the R programming environment. The package provides methods for fitting a

CT when the response is ordinal. We note that another R package, **party** (Hothorn *et al.* 2009), can also be used to derive an ordinal conditional inference tree, where the variable selected for splitting a given node is determined using an inferential test (Hothorn *et al.* 2006). These methods may prove useful when the dataset to be analyzed includes an ordinal response and the number of covariates exceeds the sample size. In such situations, traditional ordinal response methods such as proportional odds models cannot be fit. Secondary data analyses are a natural and desirable by-product from publicly available databases such as Gene Expression Omnibus. In the high-throughput genomic setting, most attention has been focused on classification algorithms for dichotomous responses. We believe analysts will find the **rpartOrdinal** useful particularly when modeling ordinal responses for high-dimensional datasets.

Acknowledgments

This research was supported by the National Institute of Library Medicine R03LM009347.

References

- Agresti AA (2002). *Categorical Data Analysis*. 2nd edition. John Wiley & Sons, Hoboken, NJ.
- Archer KJ (2010). “rpartOrdinal: An R Package for Deriving a Classification Tree for Predicting an Ordinal Response.” *Journal of Statistical Software*, **34**(7), 1–17. ISSN 1548-7660. URL <http://www.jstatsoft.org/v34/i07>.
- Archer KJ, Mas VR (2009). “Ordinal Response Prediction Using Bootstrap Aggregation, with Application to a High-throughput Methylation Dataset.” *Statistics in Medicine*, **28**, 3597–3610.
- Breiman L, Friedman JH, Olshen RA, Stone CJ (1984). *Classification and Regression Trees*. Wadsworth Statistics/Probability Series. Wadsworth Advanced Books and Software, Belmont, CA.
- Hosmer DW, Lemeshow S (2000). *Applied Logistic Regression*. 2nd edition. John Wiley & Sons, New York.
- Hothorn T, Hornik K, Strobl C, Zeileis A (2009). *party: A Laboratory for Recursive Partitioning*. R package version 0.9-999, URL <http://CRAN.R-project.org/package=party>.
- Hothorn T, Hornik K, Zeileis A (2006). “Unbiased Recursive Partitioning: A Conditional Inference Framework.” *Journal of Computational and Graphical Statistics*, **15**(3), 651–674.
- Piccarreta R (2001). “A New Measure of Nominal-Ordinal Association.” *Journal of Applied Statistics*, **28**, 107–120.
- Piccarreta R (2008). “Classification Trees for Ordinal Variables.” *Computational Statistics*, **23**, 407–427.

- R Development Core Team (2009). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0, URL <http://www.R-project.org/>.
- Steinberg D, Colla P (1997). *CART-Classification and Regression Trees*. Salford Systems, San Diego, CA.
- Steinberg D, Golovnya M (2006). *CART 6.0 User's Manual*. Salford Systems, San Diego, CA.
- Therneau TM, Atkinson EJ (1997). "An Introduction to Recursive Partitioning Using the **rpart** Routine." *Technical Report 61*, Section of Biostatistics, Mayo Clinic, Rochester. URL <http://www.mayo.edu/hsr/techrpt/61.pdf>.

Affiliation:

Kellie J. Archer
Department of Biostatistics
Virginia Commonwealth University
830 East Main Street
Richmond, Virginia, 23298-0032
E-mail: kjarcher@vcu.edu
URL: <http://www.people.vcu.edu/~kjarcher/>