CHAPTER 14

Conditional inference trees and random forests

What you will learn from this chapter:

This chapter discusses conditional inference trees and random forests. These are non-parametric tree-structure models of regression and classification that can serve as an alternative to multiple regression. They are especially useful in the presence of many high-order interactions and in situations when the sample size is small, but the number of predictors is large. You will learn how to fit such models, interpret their results and evaluate their quality. The case study that illustrates the techniques deals with three English causative constructions *make* + *V*, *cause* + *to V* and *have* + *V* and identifies the set of independent semantic variables that are important for distinguishing between the constructions.

14.1 Conditional inference trees and random forests

Conditional inference trees and random forests were introduced to linguistic analysis in a paper by Tagliamonte & Baayen (2012). **Conditional inference trees** are a method for regression and classification based on binary recursive partitioning. The latter involves several steps:

(1) the algorithm tests if any independent variables are associated with the given response variable, and chooses the variable that has the strongest association with the response.
(2) the algorithm makes a binary split in this variable, dividing the dataset into two subsets. In case of a binary predictor with values *A* and *B*, one subset will contain all observations with value *A,* and the other will contain all cases with value *B*. If a variable has more levels, one group may have values *A* and *B*, and the other may contain observations with *C*. If the variable is quantitative, the range of its values can be split into two, e.g. values from 0 to 100 can be split into two subsets: from 0 to 50 and from 51 to 100.
(3) the first two steps are repeated for each subset until there are no variables that are associated with the outcome at the pre-defined level of statistical significance. This is why the algorithm is called recursive. The result of this process can be visualized as a tree structure with binary splits forming 'branches' and 'leaves'.

Conditional inference trees offer several advantages over traditional approaches, such as the one implemented in `rpart()` in the package under the same name. First, variable selection is unbiased (the traditional method is biased towards variables with many possible splits). Second, one does not have to 'prune' (i.e. simplify) the resulting tree to avoid overfitting. Third, the algorithm also returns the *p*-values that show how confident one can be about every split.

To obtain the *p*-values, the algorithm uses **permutation**. Permutation means that the labels on the observed data points are rearranged many times, and for each rearrangement the relevant test statistic is computed. This is a way of obtaining the distribution of the test statistic under the null hypothesis of no difference, no association, etc. Next, one determines the proportion of the permutations that provide a test statistic greater than or equal to the one observed in the original data. If that proportion is smaller than some significance level, then the result is significant at that level. Permutation is similar to bootstrapping in that they both are non-parametric resampling methods, which use the same data for validation of results. However, they are not identical, since the former involves reshuffling of the labels, whereas the latter draws numerous random samples from the original sample. More details about the algorithm and available options can be found in Hothorn et al. (2006).

From many conditional trees one can grow a **random forest**. Random forests can yield the importance measure for every variable in the model averaged over many conditional trees. This measure reflects the impact of each predictor given all other independent variables.

Conditional inference trees and random forests can be particularly useful in situations of data sparseness ('small *n* large *p*', where *n* is the number of observations and *p* is the number of predictors), high-order interactions, and highly correlated predictors (Tagliamonte & Baayen 2012). In addition, recursive partitioning is non-parametric because it does not require distributional assumptions to be met. It is also considered to be robust in the presence of outliers.

14.2 Conditional inference trees and random forests of three English causative constructions

14.2.1 The data and hypotheses

To reproduce the code in this case study, you will need the following packages, which should be installed and loaded:

```
> install.packages("party")
> library(Rling); library(party)
```

The methods described in the first section will be illustrated by a case study of three causative constructions in English: *make* + *V* (e.g. *He made me believe he was an aristocrat*), *have* + *V* (e.g. *The professor had the students read Tolstoy in the original*) and *cause* + *to V* (e.g. *The crisis caused the demand for gold to grow*). According to previous research, *make* + *V* is the default causative in English with the widest scope of uses, whereas *cause* + *to V* denotes mostly physical causation, and *have* + *V* is typically used in inducive interpersonal causation (e.g. Levshina et al. 2013). These three constructions are only a part of the rich inventory of English causative constructions (see Chapter 15).

The dataset `caus` can be found in the package `Rling`. It is a data frame that contains nine causative constructions, each represented by fifty observations randomly sampled from the newspaper segment of the British National Corpus. For this case study, it is convenient to create a subset with three constructions of interest:

```
> data(caus)
> mhc <- caus[caus$Cx == "make_V"|caus$Cx == "have_V"|caus$Cx ==
"cause_toV", ]
> summary(mhc)
Cx             CrSem     CeSem      CdEv    Neg     Coref   Poss
cause_toV:50 Anim:88   Anim:105   Ment:34 No:143  No:148  No:142
have_V:50    Inanim:62 Inanim: 45 Phys:52 Yes: 7  Yes: 2  Yes: 8
make_V:50                         Soc:64
be_made_toV: 0
get_toV: 0
get_Ved: 0
(Other): 0
```

To get rid of the levels of *Cx* that have zero frequencies in the new data frame, one can use factor():

```
> mhc$Cx <- factor(mhc$Cx)
> summary(mhc$Cx)
cause_toV    have_V make_V
     50       50     50
```

Finally, the level names can be simplified:

```
> levels(mhc$Cx) <- c("cause", "have", "make")
> summary(mhc$Cx)
cause  have  make
50     50    50
```

The six other variables are explained in Table 14.1. They will be tested as predictors in conditional inference tree and random forest models, which predict the use of one of the three causative constructions specified in *Cx*.

Table 14.1 Semantic and syntactic variables

Variable	Values	Examples
CrSem (semantics of the Causer)	Animate	*He made me laugh.*
	Inanimate	*His joke made me laugh.*
CeSem (semantics of the Causee)	Animate	*She made him leave.*
	Inanimate	*The crisis caused the demand for gold to grow.*
CdEv (semantics of the caused event)	Mental	*She made him believe her.*
	Physical	*She made him leave.*
	Social	*The teacher had the students play a game.*
Neg (negation)	Yes	*She didn't make him leave.*
	No	*She made him leave.*
Coref (coreferentiality between the Causer and other participants)	Yes	*He made himself think more clearly.*
	No	*She made him leave.*
Poss (markers of possession between the Causer and other participants)	Yes	*She had her house elves do the dishes.*
	No	*She made him leave.*

14.2.2 Fitting a conditional inference tree model

Conditional inference trees are implemented as the `ctree()` algorithm in the `party` package. Although the dataset contains only categorical predictors, note that any kind of variables can be used. Before fitting a model, one should specify any random number for random number generation. The function `set.seed()` allows one to obtain a reproducible random result if one specifies the same number.

```
> set.seed(129)
> mhc.ctree <- ctree(Cx ~ CrSem + CeSem + CdEv + Neg + Coref + Poss,
data = mhc)
> plot(mhc.ctree)
```

Figure 14.1 presents the tree with all possible splits that are significant at the level of 0.05. The ovals contain the names of the variables selected for the best split, as well as the corresponding *p*-values. The levels of the variables are specified on the 'branches'. The bar plots at the bottom ('leaves') show the proportions of *make*, *have* and *cause* in each end node ('bin'), which contains all observations with a given combination of features. The number of observations in each bin is shown in parentheses above the boxes.

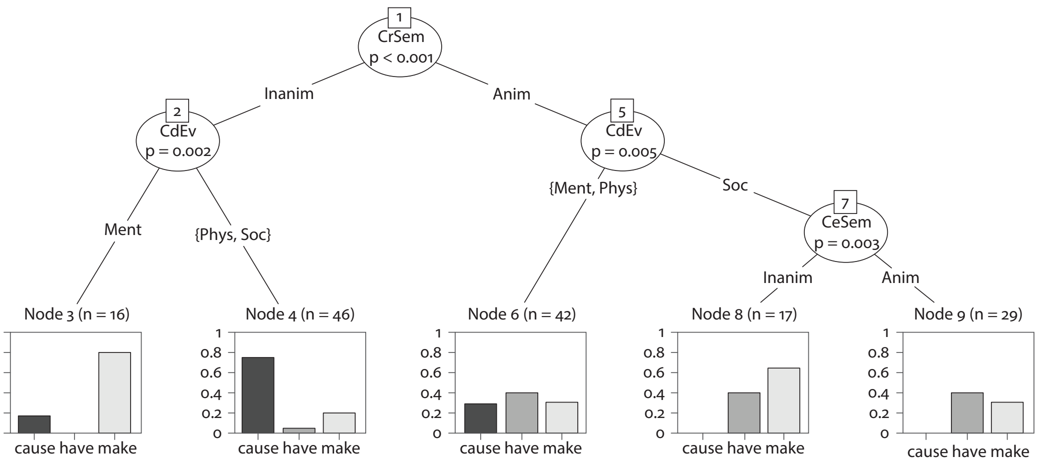

Figure 14.1. Conditional inference tree of *make + V*, *have + V* and *cause + to V*

Control parameters in conditional inference trees

It is possible to change the parameters of the algorithm, such as the level of significance, the minimum number of observations in a bin and other things. For example, if you wish to run a Monte-Carlo simulation with the significance level of 0.01 and 15 as the minimum number of observations in the bin, you can use the following code:

```
> mhc.ctree1 <- ctree(Cx ~ CrSem + CeSem + CdEv + Neg + Coref +
Poss, data = mhc, controls = ctree_control(testtype = "MonteCarlo",
mincriterion = 0.99, minbucket = 15))
```

See more information in `help(ctree_control)`.

The first split at the top divides animate and inanimate Causers (Node 1). The left branch with inanimate Causers is split into two branches (Node 2): contexts with mental caused events (Node 3) and physical or social caused events (Node 4). Node 3 contains 16 observations. Such contexts can be characterized as affective causation (see Chapter 12 on this and other types of causative situations). Most of them contain *make*, for example:

(1) *Great course … I learned a lot and it really **made** me think about the way I communicate with people.*

Node 4, which contains 46 observations, has many instances of physical causation, which explains why *cause* is the most frequent verb in this group. Consider an example:

(2) *It causes the plant to rot.*

Let us now explore the right branch. The next split is found at Node 5, which separates the observations with mental and physical caused events (Node 6) from those with social caused events on the right. The latter are then split at Node 7 into those with inanimate Causees (Node 8) and animate Causees (Node 9). The bin at Node 9 contains 29 observations. It represents inducive interpersonal causation, coded predominantly with the *have*-construction, such as in (3):

(3) *She **had** us use the lecture room for an in-class discussion.*

Node 8, in contrast, contains mostly volitional causation represented by 17 observations. In this group, *make* is the predominant causative auxiliary:

(4) *Everyone **made** it happen.*

Finally, Node 6 is a mixed category with different causation types represented nearly equally by all three verbs, for instance:

(5) *I have sometimes **made** him laugh heartily.*

The results corroborate previous findings. One can conclude that *cause* + *to V* is associated with physical causation, whereas *have* + *V* typically expresses inducive causation. The default construction *make* + *V* is represented in every bin, although it seems to be particularly 'favoured' in situations of volitional and affective causation. In general, only those variables that reflect the semantics of the participants and that of the caused event seem to be useful for discriminating between the three constructions.

Does the tree represent the data well? To answer this question, one can cross-tabulate the predicted probabilities with the observed outcomes:

```
> table(predict(mhc.ctree), mhc$Cx)

              cause       have     make
cause         35          2        9
have          12          42       17
make          3           6        24
> (35 + 42 + 24)/150
[1] 0.6733333
```

If the numbers on the table diagonal are added up, the resulting number is the number of correct predictions. Here the correct predictions are made for 0.673, or 67.3% of the total 150 observations. If the algorithm simply assigned the verbs randomly, it would be correct in 33.3% of the cases. So, the model offers some improvement beyond that baseline.

To ensure that the results are stable, it is recommended to run the analysis several times with different random number seeds and compare the resulting models.

14.2.3 Random forests

To create a random forest, one can use the function `cforest()` from the same package:

```
> set.seed(35)
> mhc.rf <- cforest(Cx ~ CrSem + CeSem + CdEv + Neg + Coref + Poss,
data = mhc, controls = cforest_unbiased(ntree = 1000, mtry = 2))
```

The argument `ntree = 1000` tells R to grow a large forest with 1000 trees. It is recommended to define `mtry` (i.e. the number of randomly preselected predictors at each split) as the square root of the number of predictors, but one should try several different values.

Next, the measures of variable importance are computed. Note that this procedure is computationally intensive and you may have to wait for some time. Since the algorithm is based on permutation, your results will be slightly different every time you execute the code, unless you use `set.seed()` with the same number every time you compute the measures.

```
> mhc.varimp <- varimp(mhc.rf, conditional = TRUE)
> round(mhc.varimp, 3)
CrSem  CeSem  CdEv   Neg    Coref  Poss
0.098  0.043  0.045  0.000  0.000  0.000
```

The variable importance scores demonstrate that the semantics of the Causer is the most important predictor (0.098), followed by the semantic class of the caused event (0.045) and that of the Causee (0.043). Your results may slightly differ because the procedure is based on random reshuffling of the data. These are also the variables that were responsible for the splits in the conditional inference tree. Negation, coreferentiality and possession relations do not seem to have any discriminatory power. The impact of variables can be visualized in a dot chart, as shown in Figure 14.2:

```
> dotchart(sort(mhc.varimp), main = "Conditional importance of
variables")
```

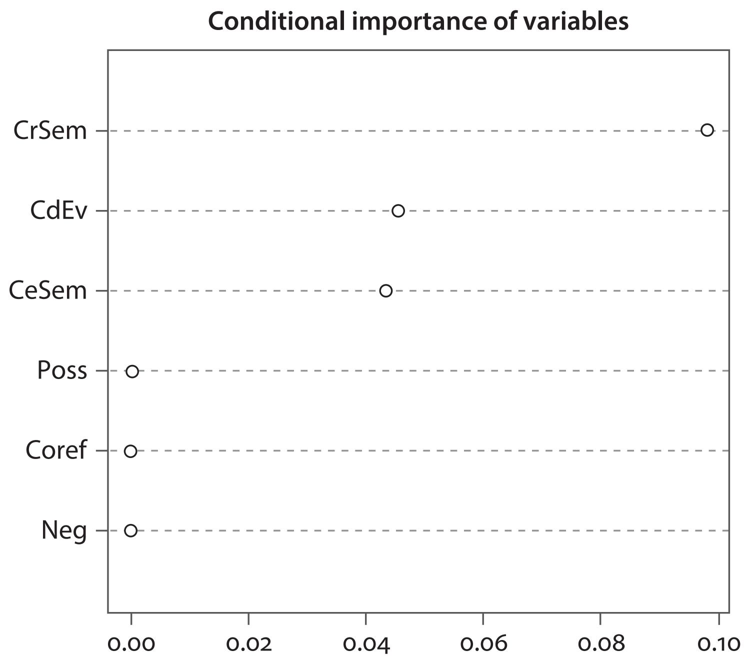

Figure 14.2. Dotchart of conditional variable importance

If a variable is irrelevant, its importance values will vary around the zero. Sometimes it is difficult to say which variable is relevant and which is not. According to a rule of thumb, the cut-off value is the absolute importance value of the variable with the smallest score.

To obtain an idea of how well the model fits the data, one can again compute the accuracy measure:

```
> table(predict(mhc.rf), mhc$Cx)

       cause  have  make
cause  39     2     9
have   8      48    28
make   3      0     13
> (39 + 48 + 13)/150
[1] 0.6666667
```

The accuracy is twice as large as the one that one could expect by chance. Note, however, that the model predicts *have* far too often (8 + 48 + 28 = 84 times), and 'underpredicts' *make* (3 + 13 = 16 times). There is definitely room for improvement, e.g. by trying larger values of `mtry`.

How to compute *C*-index for binary response

If your response variable is binary, you can obtain the *C*-index to evaluate the goodness of fit of your conditional inference tree or random forest model (see Chapter 12). It can be computed with the help of the function `somers2()` from the package `Hmisc`. For a conditional inference tree model, this can be done as follows:

```
# do not run; provided only as an example
> your.ctree.pred <- unlist(treeresponse(your.ctree))[c(FALSE,
TRUE)]
> somers2(your.ctree.pred, as.numeric(yourdata$Outcome) - 1)
```

To compute the *C*-index for a random forest model, replace the argument `your.ctree` with your random forest object.

Similar to conditional inference trees, it is recommended to rerun random forests several times with different random number seeds and compare the results.

14.3 Summary

This chapter has introduced conditional inference trees and random forests, two recent non-parametric techniques that can be used when regression modelling is problematic (complex interactions, too few observations in comparison with the number of variables, etc.). You have learnt how to fit, evaluate and interpret such models. This chapter concludes the large section on statistics for hypothesis testing. The final large part of this book will discuss exploratory methods that can be used for detecting structure in multivariate data.

Writing up your results

Since the techniques are relatively novel, there seem to be no rules for reporting the results. It is recommended to use a graphical representation and report the goodness-of-fit statistics.

More on trees and forests

See another example of conditional inference trees and random forests in Tagliamonte & Baayen (2012). For more information about traditional binary recursive partitioning methods, see Crawley (2007: Ch. 21).