Decision tree methods for psychological assessment

Workshop Presented by Marjolein Fokkema at ECPA15, Brussels, July 7 2019

Contents

1. Introduction

2. Split Selection

3. Regression Tree Example: Predicting Beck Depression Inventory Scores

4. Classification Tree Example: Predicting Chronic Trajectories of Anxiety and Depression

5. Model-Based Recursive Partitioning

6. Model-Based Tree Example: Subgroups With Differential Treatment Effects

7. Mixed-Effects Model Trees

8. Mixed-Effects Model Tree Example: Subgroups With Differential Treatment Effects

9. Mixed-Effects Model Tree Example: Subgroups With Different Reading Skill Trajectories

10. Further Reading and Methods

11. References

1. Introduction

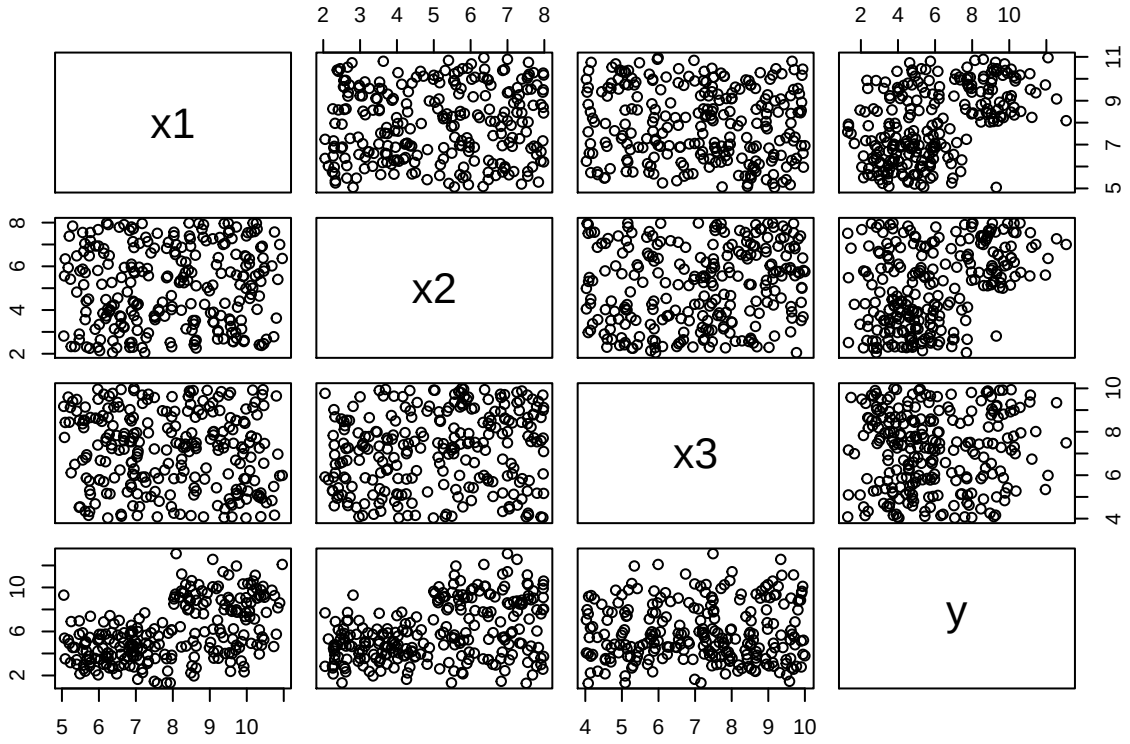
For a first introduction to decision tree methods, we use a toy dataset containing three continuous variables: predictor variables x_1, x_2 and x_3, and response variable y. The dataset is available as the file "toy data.txt". After downloading it to the current working directory, we can load it into **R** using the `read.table()` function:

```
toy_data <- read.table("toy data.txt")
nrow(toy_data)
```

```
## [1] 250
```

We first inspect the data by plotting it:

```
plot(toy_data)
```



The plots indicate that variables x_1 and x_2, but not x_3, are associated with y. There appears to be no association between x_1, x_2 and x_3. This is in line with how the data were generated.

Let's first fit a model most of us will already be familiar with: ordinary least squares (OLS) regression. In **R**, we use the `lm()` function to fit an OLS model:

```
lin_mod <- lm(y ~ x1 + x2 + x3, data = toy_data)
```

With the first argument of this function, we defined the model formula, which specifies the response, followed by the tilde symbol (`~`) and then the potential predictor variables. We also need to specify the `data` argument, containing the variables appearing in the model formula.

We can inspect the fitted model using the `summary` method:

```
summary(lin_mod)
```

```
##
## Call:
## lm(formula = y ~ x1 + x2 + x3, data = toy_data)
```

```
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.4126 -1.3613 -0.0884  1.3833  6.5964
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -2.82669    0.91784  -3.080  0.00231 **
## x1           0.74614    0.08116   9.193  < 2e-16 ***
## x2           0.49354    0.07500   6.580 2.81e-10 ***
## x3           0.04094    0.07695   0.532  0.59521
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.086 on 246 degrees of freedom
## Multiple R-squared:  0.3652, Adjusted R-squared:  0.3574
## F-statistic: 47.17 on 3 and 246 DF,  p-value: < 2.2e-16
```

The results indicate that x_1 and x_2 are positively and significantly associated with y. Using the coefficients above, if we wanted to predict the values of y for new observations, for which we know only the values of the x variables, but not the value of y, we could use the following prediction equation:

$\hat{y} = \text{-}2.827 + 0.746\ x_1 + 0.494\ x_2 + 0.041\ x_3$

In **R**, we can use the `predict` method to perform the computations (although note that these observations are not 'new', but were part of the training data):

```
predict(lin_mod, newdata = toy_data[1:8, ])
```

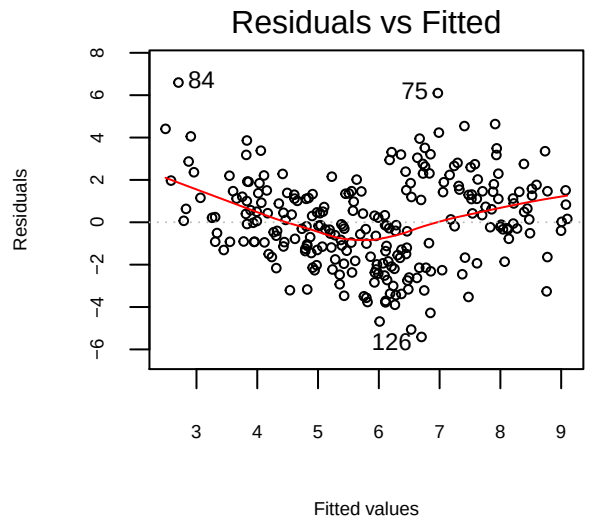
```
##        1        2        3        4        5        6        7        8
## 3.542662 6.485249 5.927980 5.432306 5.635332 6.441121 2.791059 4.822588
```

The statistical tests for evaluating the (linear) effects of the predictor variables assume the residuals to be normally distributed, which we can check through the following plot:

```
plot(lin_mod, which = 1)
```

```
## Warning in plot.window(...): "sub.cex" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "sub.cex" is not a graphical parameter
## Warning in axis(side = side, at = at, labels = labels, ...): "sub.cex" is
## not a graphical parameter
## Warning in axis(side = side, at = at, labels = labels, ...): "sub.cex" is
## not a graphical parameter
## Warning in box(...): "sub.cex" is not a graphical parameter
## Warning in title(...): "sub.cex" is not a graphical parameter
## Warning in plot.xy(xy.coords(x, y), type = type, ...): "sub.cex" is not a
## graphical parameter
```

```
## Warning in title(sub = sub.caption, ...): "sub.cex" is not a graphical
## parameter
```



The residuals are not normally distributed. For higher and lower predicted values, the residuals are higher. There are several possible causes for this, one of them being that the association between y and the predictors is not linear.

We continue to fit a regression tree, using function `ctree()` (short for *conditional inference tree*) from package **partykit**. If we have not installed that package in **R** yet, we can do that as follows:

```
install.packages("partykit")
```

We only have to install the package once. After that, we can load the package into **R** as follows:

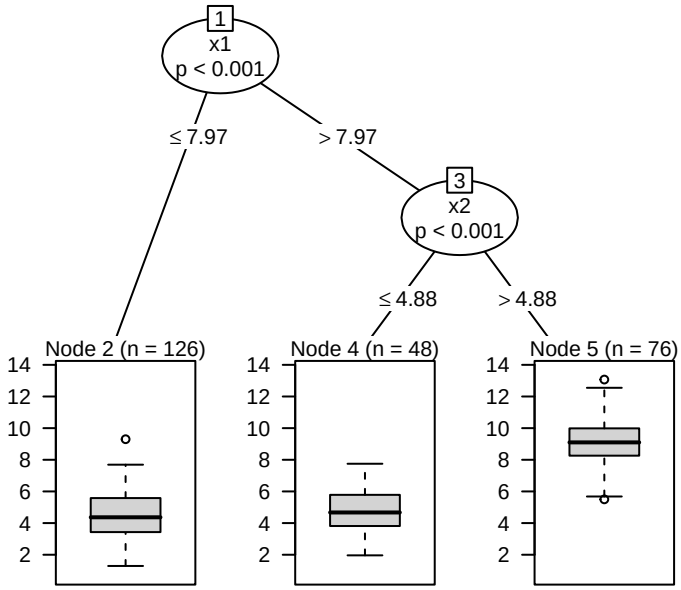
```
library("partykit")
```

To apply the `ctree()` function, we specify the `formula` and `data` arguments:

```
tree <- ctree(y ~ x1 + x2 + x3, data = toy_data)
```

We can plot the fitted tree using the `plot` method:

```
plot(tree, gp = gpar(cex = .65))
```

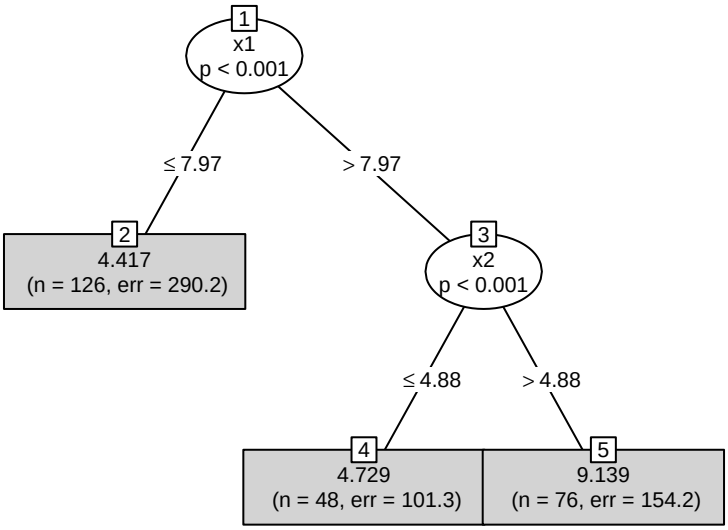


Note that by specifying `gp = gpar(cex = .65)`, we have reduced the size of the labels of nodes and edges.

The tree reveals three subgroups: a subgroup with values $x1 \leq 7.97$, which has the lowest values of y (node 2). Then two subgroups with values of $x1 > 7.97$: observations in node 4 (with values $x2 \leq 4.88$) do not seem to have strikingly larger values than those in node 2, while the observations in node 5 (with values $x2 > 4.88$) shows strikingly higher values values of y. This pattern seems to indicate an interaction effect, where the effect of x_2 is dependent on the value of x_1.

To obtain a prediction of the value of y for a new observation, we would 'drop' it down the tree, and based on its values on the predictor variables, it would end up in one of the terminal node. The predicted value is the mean of y of the training observations in that terminal node. We can request a plot which shows the predicted values in each of the terminal nodes as follows:

```
plot(tree, gp = gpar(cex = .65), type = "simple")
```



Alternatively, we can use the `predict` method in **R**:

```
predict(tree, newdata = toy_data[1:8, ])
```

```
##        1        2        3        4        5        6        7        8
## 4.416905 4.728542 4.416905 4.416905 4.728542 9.138816 4.416905 4.416905
```

2. Split selection

In the plots above, we saw that every split in the plotted tree has a p value. These p values guide the splitting (or *partitioning*) process: They test the null hypothesis of independence between a potential partitioning variable and the response. The lower the p value, the stronger the evidence in favor of an association between the partitioning variable and the response.

More specifically, a conditional inference tree (ctree) is created as follows:

1. **Variable selection**: In the current node, the association between the response and each of the potential partitioning variables is tested. If at least one of the potential partitioning variables has a p value equal to or lower than a pre-specified value of α, the variable with the lowest p value is selected for splitting.
2. **Split point selection**: The value of the selected splitting variable that minimizes the loss criterion in the two resulting terminal nodes is selected for splitting.
3. Repeat steps 1 and 2 in the two resulting nodes.

This approach provides two main advantages, compared to other decision-tree methods:

- It provides unbiased variable selection. Other decision-tree methods, like the classification and regression tree (CART) algorithm of Breiman, Friedman, Olshen and Stone (1984), select the splitting variable and split point in one step, which yields a selection bias towards variables with a larger number of possible splitting values. See Hothorn, Hornik and Zeileis (2006) and Loh and Shih (1997) for a more detailed discussion.
- It provides a natural stopping criterion: When none of the partitioning variables in the current node have a p value $\geq \alpha$ (which equals .05, by default, but other values can be specified), splitting is halted.

The tests and methods used for variable and split-point selection are discussed in more detail in Hothorn, Hornik and Zeileis (2006) and Hothorn, Hornik, Van De Wiel and Zeileis (2006). A more detailed description can also be found in the `ctree` vignette (i.e., long-form **R** package documentation; it can be opened by typing `vignette("ctree")`). For the current introductory workshop, it suffices to say that these tests assume the observations are independent, but do not involve normality assumptions.

We can use function `sctest.constparty` to extract the test statistics and p values computed in each node:

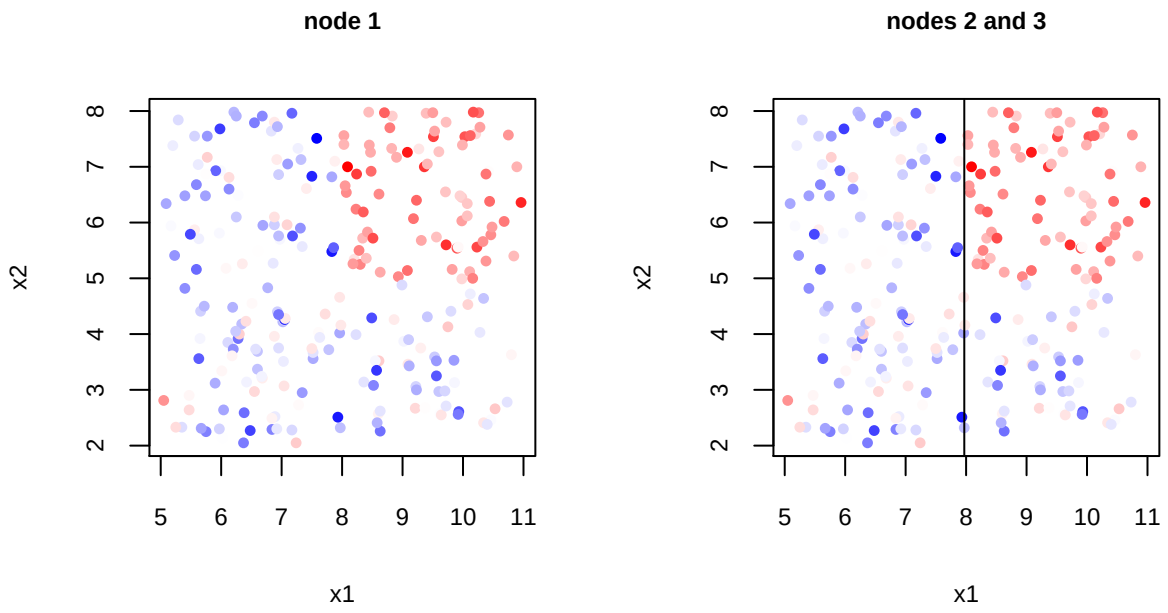
```
sctest.constparty(tree, node = 1)
```

```
##                     x1           x2        x3
## statistic 6.120936e+01 3.644008e+01 0.3462214
```

```
## p.value    1.539507e-14 4.722965e-09 0.9126252
```

Here we obtained the test statistics computed for the first (root) node, which contains all observations in the data set. Based on all observations, x_1 and x_2 have obtained relatively high test statistics, yielding very small p values. Variable x_3 obtained a low value for the test statistic and a p value close to one, indicating that x_3 and y are not associated. Variable x_1 will be used for splitting, as it has the lowest p value.

We can graphically depict the splitting procedure. First, we plot the values of x_1, x_2 and y (we ignore variable x_3, because it has no association with y):



In the left panel (node 1), all observations are in the same (root) node. The colors of the dots represent the values of y: higher values are (more) red, lower values are (more) blue. The value of $x_1 = 7.97$ was selected for splitting, as it best separates observations with lower and higher values of y.

The first split created two nodes, or subgroups: node 2 and node 3. Now the statistical tests are performed in each of these resulting nodes:

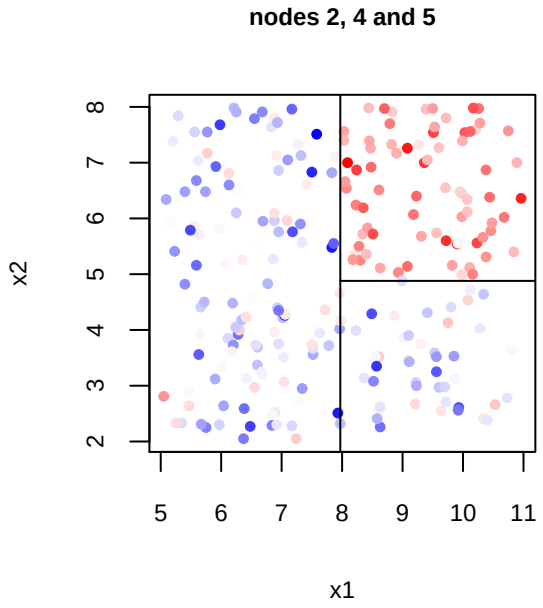
```
sctest.constparty(tree, node = c(2, 3))
```

```
## $`2`
##                  x1        x2        x3
## statistic 0.9964731 2.6553082 1.3130833
## p.value   0.6830162 0.2787598 0.5812176
##
## $`3`
```

```
##                    x1           x2        x3
## statistic 0.00330644 6.685766e+01 3.9356145
## p.value   0.99990359 8.755092e-16 0.1352228
```

In node 2, none of the partitioning variables obtained a p value $\leq .05$ and therefore this node is not splitted further.

In node 3, only x_2 is significantly associated with y, as indicated by the p value. Thus, the observations in this node are now split using variable x_2, into nodes 4 and 5:



The value of 4.88 was found to best separate observations with lower and higher values of y.

We can again request the test results for each of the partitioning variables in the resulting two nodes:

```
sctest.constparty(tree, node = c(4, 5))
```

```
## $`4`
##                    x1        x2         x3
## statistic 1.2685714 0.2389616 0.05307592
## p.value   0.5948333 0.9472475 0.99395102
##
## $`5`
##                    x1         x2        x3
## statistic 0.4845901 0.07074638 1.6192195
## p.value   0.8644809 0.99077245 0.4941207
```

None of the partitioning variables obtained a p value $\leq .05$ in any of the current terminal nodes. Thus,

splitting is halted and the tree with terminal nodes 2, 4 and 5 is the final tree. The plots above (hopefully) illustrate that the partitioning is recursive: every split is conditional on earlier splits. Also, they show how the recursive partitioning separates the space spanned by the predictor variables into *rectangular* areas, which is a distinctive property of decision-tree methods.

3. Regression Tree Example: Predicting Beck Depression Inventory Scores

For this example, we make use of a dataset from a study by Carrillo et al. (2001), who examined the association between big five personality characteristics (as measured by the NEO-PI-R) and depression (as measured by the Beck Depression Inventory). These data are available as the file "carrillo data.txt"; we read it into **R** as follows:

```
carrillo <- read.table("carrillo data.txt")
names(carrillo)
```

```
##  [1] "n1"      "n2"      "n3"      "n4"      "n5"      "n6"      "ntot"
##  [8] "e1"      "e2"      "e3"      "e4"      "e5"      "e6"      "etot"
## [15] "open1"   "open2"   "open3"   "open4"   "open6"   "opentot" "altot"
## [22] "contot"  "bdi"     "sexo"    "edad"    "open5"
```

```
nrow(carrillo)
```

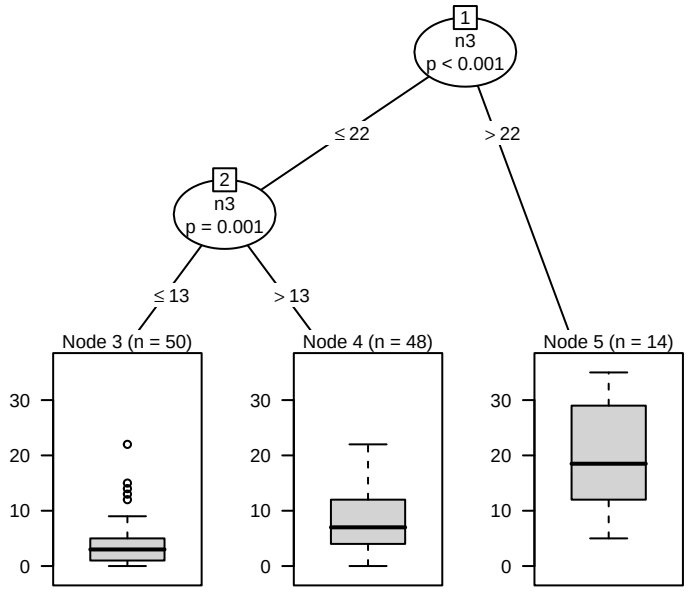
```
## [1] 112
```

The dataset contains facet and subscale scores for the neuroticism, extraversion and openness subscales. Furthermore, it contains subscale scores for altruism and conscientiousness, as well as an indicator for gender (`sexo`) and age (`edad`). The data consists of 112 observations. We fit a tree for predicting the BDI scores using function `ctree()`:

```
bdi_tree <- ctree(bdi ~ ., data = carrillo)
```

Because there are many possible partitioning variables, we used the dot after the vertical bar (`.`) as short-hand notation to specify that all remaining variables in the dataset should be used as possible partitioning variables. Next, we plot the tree:

```
plot(bdi_tree, gp = gpar(cex = .6))
```



Only one variable was found predictive of the BDI scores: `n3`, a neuroticism facet scales which (unsurprisingly) measures Depression. The higher the value of `n3`, the higher the value of `bdi`. The remaining personality scores, age and gender do not further contribute to predicting BDI scores, over and above the n3 facet.

If we want to obtain the exact subgroup means, we can print the fitted tree:

```
bdi_tree
```

```
##
## Model formula:
## bdi ~ n1 + n2 + n3 + n4 + n5 + n6 + ntot + e1 + e2 + e3 + e4 +
##     e5 + e6 + etot + open1 + open2 + open3 + open4 + open6 +
##     opentot + altot + contot + sexo + edad + open5
##
## Fitted party:
## [1] root
## |   [2] n3 <= 22
## |   |   [3] n3 <= 13: 3.960 (n = 50, err = 1015.9)
## |   |   [4] n3 > 13: 8.458 (n = 48, err = 1575.9)
## |   [5] n3 > 22: 20.643 (n = 14, err = 1167.2)
##
## Number of inner nodes:    2
## Number of terminal nodes: 3
```

4. Classification Tree Example: Predicting Chronic Trajectories of Anxiety and Depression

In this example, we will predict a multivariate categorical outcome: the presence of anxiety, and the presence of depression. I have created a dataset that mimics the data used in Penninx et al. (2011), who aimed to predict chronic anxiety and depressive disorder trajectories among respondents with a current disorder. In that study, a chronic trajectory was operationalized as the presence of a disorder at two-year follow-up. Originally, logistic regression analyses with 20 potential predictor variables were performed. I generated the dataset so as to contain the same predictor and response variables, with very similar distributions. A detailed description of the original data can be found in Penninx et al. (2011) or Fokkema, Smits, Kelderman & Penninx (2015).

```
NESDA <- read.table("NESDA_mimic.txt")
dim(NESDA)
```

```
## [1] 1100   22
```

```
names(NESDA)
```

```
##  [1] "dep"        "anx"        "disType"    "Sexe"       "Age"
##  [6] "aedu"       "IDS"        "BAI"        "FQ"         "LCImax"
## [11] "pedigree"   "alcohol"    "bTypeDep"   "bSocPhob"   "bGAD"
## [16] "bPanic"     "bAgo"       "AO"         "RemDis"     "sample"
## [21] "ADuse"      "PsychTreat"
```

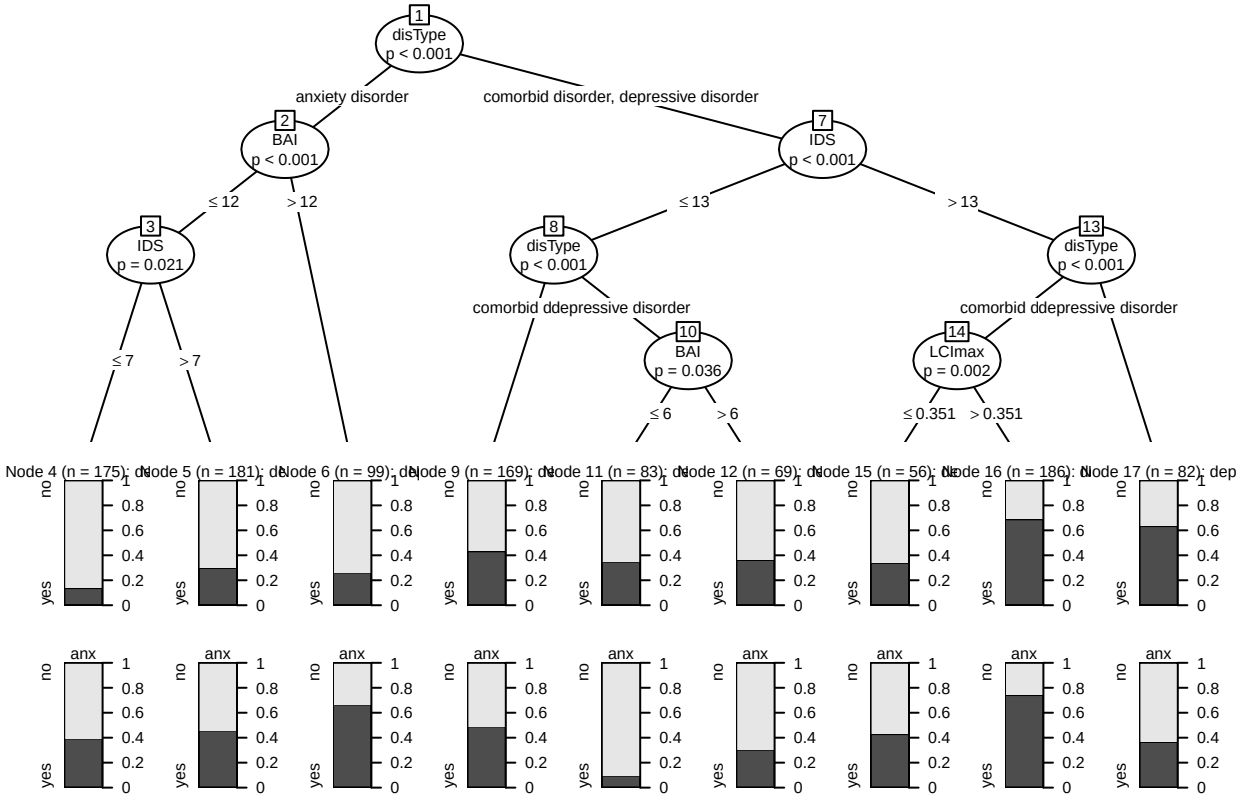
The dataset contains 1100 observations and 22 variables. The `dep` and `anx` variables represent the response variables: i.e., presence of depressive and anxiety diagnoses at two-year follow-up.

The potential predictor variables are: `disType`, which reflects baseline psychiatric status and distinguishes between depression, anxiety and comorbid depression and anxiety. `Sexe` and `Age` are self explanatory. `aedu` reflects the years of completed education. `IDS` reflects the score on a measure of depressive symptoms, `BAI` and `FQ` reflect the scores on measures of anxiety symptoms. `LCImax` reflects the proportion of time in which symptoms of anxiety and depression were present in the four years prior to baseline. `pedigree` is an indicator for whether the respondent has a first-degree relative with anxiety or depressive disorder. `alcohol` reflects the presence of an alcohol disorder diagnosis. `bTypeDep` reflects the type of depressive disorder at baseline: no depressive disorder, first episode major depressive disorder (MDD), recurrent MDD, and dysthymia. `bSocPhob`, `bGAD`, `bPanic`, and `bAgo` reflect the presence of anxiety disorders at baseline: social phobia, generalized anxiety disorder, panic disorder and agoraphobia without panic disorder, respectively. `AO` represents the age of onset of the index disorder. `sample` is an indicator for whether respondents were originally recruited from primary

care, specialized mental health care, or from the general population. `PsychTreat` and `ADuse` are indicators for whether the respondent was receiving psychological treatment and/or anti-depressant medication.

Our aim is to predict both the presence of anxiety and depressive disorders, and we will fit a single tree for both response variables simultaneously. Not all tree-growing functions in **R** support multivariate responses, but `ctree()` does. When using `ctree()`, we specify a multivariate response by combining the response variables with a `+` sign:

```
NESDA_tree <- ctree(dep + anx ~ ., data = NESDA)
plot(NESDA_tree, gp = gpar(cex = .5))
```



The results reveal 8 subgroups with different probabilities of having anxiety or depressive disorder. The upper row of barplots represent the probability of a depressive disorder at two-year follow-up. The bottom row represents the probability of anxiety disorder. In general, higher anxiety or depressive symptom scores are associated with higher probability of a chronic trajectory.

The lowest probability for anxiety as well as depression are observed in node 11: for respondents with only a depressive disorder at baseline, and relatively low depressive (IDS) and anxiety (BAI) symptoms. The highest probability for anxiety as well as depression are observed in node 16: for respondents with a comorbid

disorder at baseline, relatively higher depressive symptoms, and symptoms of anxiety and/or depression present 35% of the time or more.

5. Model-Based Recursive Partitioning

In the preceding examples, we focused on regression and classification problems, where there was one (or more) dependent variables, and one or more predictor (or independent, or partitioning) variables. The trees had so-called *constant fits* in the terminal nodes: that is, the predicted values is the same (i.e., a constant) for all observations in the same terminal node.

We now continue with trees which have parametric models instead of constant fits in the terminal node. These trees are also known as model-based recursive partitioning (MOB), a methodology developed by Zeileis, Hothorn and Hornik (2008). The underlying rationale of MOB is that there is a (relatively simple) global parametric model. For example, we have repeated measurements of a response variable y, and we want to model the effect of time (x) on the response. Or, we have a data from a randomized clinical trial, and we want to model the effect of treatment (x) on the response variable y. Often, we know that such a simple global model may not fit all observations well. With MOB, we aim to partition the observations with respect to additional covariates, and find better-fitting models in each cell of the partition. In other words, MOB finds subgroups which have different values of the parameter estimates. For example, when we model trajectories or growth over time, there may be subgroups that differ in terms of the rate of growth over time, e.g., some subgroups may show a decline while other subgroups show an increase, or some subgroups may have higher or lower initial values.

With MOB, the splitting is performed in a similar manner as with conditional inference trees (for an in-depth description, see Zeileis, Hothorn & Hornik, 2008):

1. In the current node, the parametric model is fit once to all observations. Next, a *parameter stability test* is performed for each of the partitioning variables. If at least one of the potential partitioning variables has a p value for the parameter stability test $\leq \alpha$, the variable with the lowest p value is selected for splitting.

2. The split point is selected, that yields optimal fit when the parametric model is fitted in each of the resulting two nodes.

3. Steps 1 and 2 are repeated in the two resulting nodes.

So, with model-based recursive partitioning, we have three types of variable: a response and one or more predictor variables for the parametric model, and the partitioning variables (additional covariates). An in-depth discussion of the parameter stability tests is outside the scope of this paper, but can be found in Zeileis, Hothorn and Hornik (2008)

We can perform model-based recursive partitioning with (amongst others) functions `mob()`, `lmtree()` and `glmtree()` from package **partykit**. When the simple parametric model is a (multiple) linear regression, it is easiest to use function `lmtree()`. When the simple parametric model is a (multiple) logistic regression, it is easiest to use function `glmtree()`.

6. Model-Based Tree Example: Subgroups With Differential Treatment Effects

For this example, we will make use of artificial data modelled after the Stratified Medicine Approaches foR Treatment Selection (SMART) prediction tournament, from the Improving Access to Psychological Therapies (IAPT) project (Lucock et al., 2017). The SMART data contains data from patients receiving mental-health services in the Northern UK. Patients were (non-randomly) assigned to low intensity treatment (e.g., guided self-help, computerized cognitive behavior therapy) or high intensity treatment (e.g., face-to-face psychological therapies). The aim of the SMART tournament was to identify patients who would benefit most from HI vs. LI treatment. Because I do not own the data, I have created an artificial dataset which mimics the original data, and is available in the file "SMART mimic data.txt".

```
SMART <- read.table("SMART mimic data.txt")
dim(SMART)
```

```
## [1] 1500   13
```

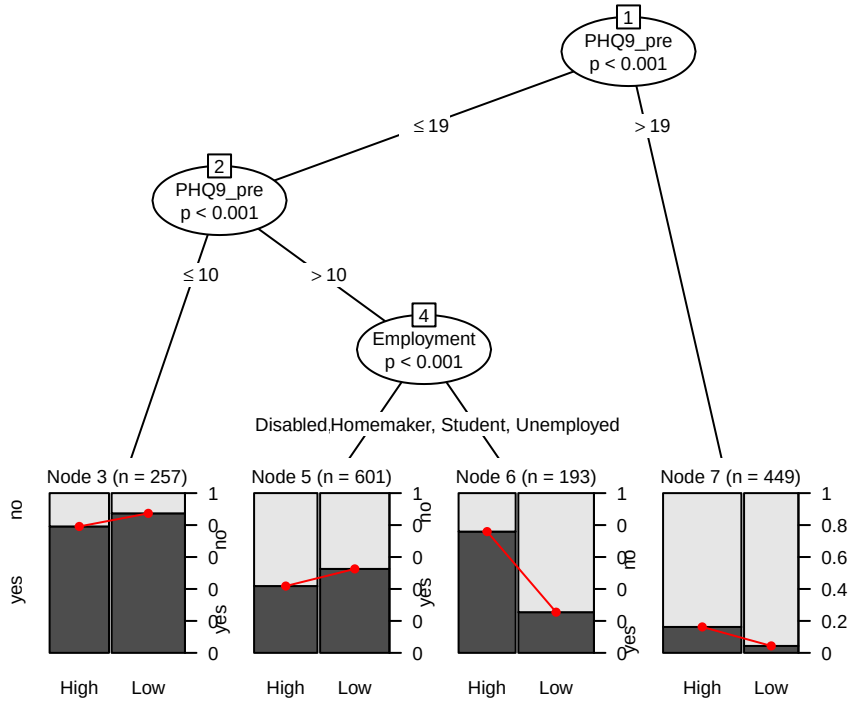
```
names(SMART)
```

```
##  [1] "recovered"  "Treatment"  "Age"        "Gender"     "Ethnicity"
##  [6] "Diagnosis"  "Employment" "Disability" "PHQ9_pre"   "GAD7_pre"
## [11] "WSAS_pre"   "Medication" "center"
```

The dataset contains 1,500 observations and 13 variables: A binary response variable (`recovered`) and an indicator for treatment type (`Treatment`). Furthermore, as potential partitioning variables, there are `Age`, `PHQ9_pre` (baseline depression severity), `GAD7_pre` (baseline symptoms of generalized anxiety disroder) and `WSAS_pre` (baseline work and social adjustment), `Gender`, `Ethnicity`, `Diagnosis`, `Employment`, `Disability` and `Medication`. Finally, there is an indicator for treatment center (`center`), which we will use later in the multilevel analyses).

We will fit a tree to detect predictors and moderators of treatment outcome. We therefore regress the response on the treatment indicator (i.e., high vs. low intensity treatment): `recovered ~ Treatment`. Then we specify the possible partitioning variables, after a vertical bar (`|`). Because the response variable is binary, we use `glmertree()` function and specify `family = binomial`:

```
SMART_t <- glmtree(recovered ~ Treatment | Age + PHQ9_pre + GAD7_pre +
                     WSAS_pre + Gender + Ethnicity + Diagnosis +
                     Employment + Disability + Medication,
                   data = SMART, family = binomial)
plot(SMART_t, which = "tree", gp = gpar(cex = .6))
```




```
SMART_t
```

```
## Generalized linear model tree (family: binomial)
##
## Model formula:
## recovered ~ Treatment | Age + PHQ9_pre + GAD7_pre + WSAS_pre +
##     Gender + Ethnicity + Diagnosis + Employment + Disability +
##     Medication
##
## Fitted party:
## [1] root
## |   [2] PHQ9_pre <= 19
## |   |   [3] PHQ9_pre <= 10: n = 257
## |   |         (Intercept) TreatmentLow
## |   |           1.3328057    0.5971041
## |   |   [4] PHQ9_pre > 10
## |   |   |   [5] Employment in Disabled, Employed, Retired: n = 601
## |   |   |         (Intercept) TreatmentLow
## |   |   |          -0.3305021    0.4325012
## |   |   |   [6] Employment in Homemaker, Student, Unemployed: n = 193
## |   |   |         (Intercept) TreatmentLow
## |   |   |            1.147402    -2.221917
## |   [7] PHQ9_pre > 19: n = 449
## |         (Intercept) TreatmentLow
## |           -1.641477    -1.449565
##
## Number of inner nodes:    3
## Number of terminal nodes: 4
## Number of parameters per node: 2
```

```
## Objective function (negative log-likelihood): 783.841
```

Note that the plotted tree has a similar format as the trees we created using function `ctree()`: Every inner node shows the variable selected for splitting, with its corresponding p value. This p value is the result of the parameter stability test for the selected splitting variable.

In the terminal nodes of the plotted tree, the probability of recovery is depicted. The width of the bars reflect the number of observations in a node, within each of the treatment groups. Thus, node 3 contains somewhat more observations in the low intensity than in the high intensity treatment condition.

The coefficients in the printed results should be interpreted as coefficients from a logistic regression. Thus, in node 3, the `(Intercept)` shows that patients in that node, receiving high intensity treatment, have a probability of $\frac{e^{1.333}}{1+e^{1.333}} = .791$ to recover. Patients in that same node, receiving low intensity treatment have a higher probability of $\frac{e^{1.333+0.597}}{1+e^{1.333+0.597}} = .873$ to recover. Although this result may seem counter intuitive (we would expect high intensity treatment to be more effective), we should remember that patients were not randomly assigned to treatment, which may explain the less favorable results for high intensity treatment, as these patients may have had characteristics which are not coded in the dataset, but that led to them being assigned to the high intensity treatment.

The left-most node shows that patients with PHQ9 scores ≤ 10 at the start of treatment are likely to recover over the course of treatment, and even seem somewhat more likely to recover with low intensity treatment than high intensity treatment, but this difference seems negligible.

Furthermore, for patients with PHQ9 scores > 10 but ≤ 19, `Employment` is an important predictor of treatment efficacy: For homemakers, students, and unemployed persons, high intensity treatment seems much more effective. For those patients who are employed, disabled or retired, the low intensity treatment seems somewhat more effective, but again the difference between the conditions seems small.

Finally, for patients with high PHQ9 scores (i.e., > 19) at the start of treatment, recovery is unlikely, but high intensity treatment may be slightly more effective than low intensity treatment.

7. Mixed-Effects Model Trees

As mentioned earlier, the tests used for selecting the splitting variables (as any statistical test) assume the observations in the dataset are independent. However, in psychological research we often encounter *nested* data, where lower-level observations are nested within higher-level units. For example, in longitudinal datasets, observations are nested within subjects; in educational research, students can be nested within classrooms; in randomized clinical trails, patients can be nested within hospitals or research centers. In such nested designs, observations within the same higher-level unit are correlated, i.e., dependent. In model-based recursive partitioning, we can account for this dependence in two ways:

- Estimation of random effects, so as to account for the dependence between observations from the same higher-level unit, and/or
- Accounting for the nested structure in computing the parameter stability tests, if the possible partitioning variables are not measured at the observation level, but at a higher level. E.g., when possible partitioning variables are characteristics of the class(room) and not of the student, or when they are time-invariant covariates.

In Fokkema, Smits, Zeileis, Hothorn and Kelderman (2018), we developed the generalized linear mixed-effects model tree (GLMM tree) algorithm, which allows for estimating (G)LM trees, while estimating and accounting for random effects. A detailed explanation of how GLMM trees are estimated is outside the scope of this workshop, but can be found in Fokkema et al. (2018). Here, we focus on the practical aspect of fitting GLMM trees in **R**, which can be done using package **glmertree**.

If we have not installed that package in **R** yet, we can do that as follows:

```
install.packages("glmertree")
```

We only have to install the package once. After that, we can load the package into **R** as follows:

```
library("glmertree")
```

Specification of the model formula is similar to that of function `mob()`, `glmtree()` and `lmtree()`. Above, we saw that the formulas for `glmtree()` have the following shape:

```
response ~ predictor variable(s) | potential partitioning variables
```

The formulas for `glmertree()` also specify the random effects. The random effects are specified between the predictor variable(s) and the potential partitioning variables, separated by vertical bars:

```
response ~ predictor variable(s) | random effects | potential partitioning variables
```

The random effects are specified as they would be for traditional GLMMs, fitted with functions `lmer()` and `glmer()` from **R** package **lme4**.

8. Mixed-Effects Model Tree Example: Subgroups With Differential Treatment Effects

In analysis of the SMART data above, we ignored the indicator for treatment center (`center`). It is likely that the scores of patients within the same treatment center are correlated, because they are from the same geographical area, visit the same clinicians, and/or receive very similar treatments. We can account for this by specifying a random intercept term for `center`.

To include the random intercept term in the mixed-effects model tree, we take the same formula as with the `glmtree()` function above, but now also specify a random intercept with respect to `center`. If we specify a random intercept with respect to a single indicator, we only need to specify the indicator. Thus, below we type `center` as short-hand for `(1|center)`:

```
SMART_t2 <- glmertree(recovered ~ Treatment | center | Age + PHQ9_pre +
                        GAD7_pre + WSAS_pre + Gender + Ethnicity +
                        Diagnosis + Employment + Disability + Medication,
                      data = SMART, family = binomial)
```

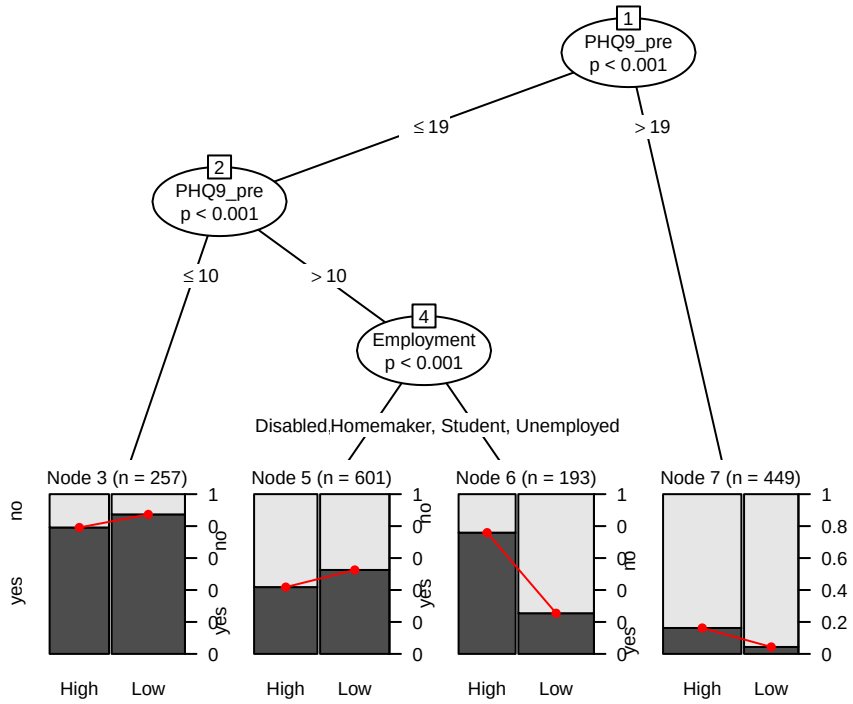
We can print and plot the results as follows:

```
SMART_t2$tree
```

```
## Generalized linear model tree (family: binomial)
##
## Model formula:
## recovered ~ Treatment | Age + PHQ9_pre + GAD7_pre + WSAS_pre +
##     Gender + Ethnicity + Diagnosis + Employment + Disability +
##     Medication
##
## Fitted party:
## [1] root
## |   [2] PHQ9_pre <= 19
## |   |   [3] PHQ9_pre <= 10: n = 257
## |   |         (Intercept) TreatmentLow
## |   |           1.5255853    0.5577717
## |   |   [4] PHQ9_pre > 10
## |   |   |   [5] Employment in Disabled, Employed, Retired: n = 601
## |   |   |         (Intercept) TreatmentLow
## |   |   |          -0.2975380    0.4358343
## |   |   |   [6] Employment in Homemaker, Student, Unemployed: n = 193
## |   |   |         (Intercept) TreatmentLow
## |   |   |            1.096908    -2.359766
## |   [7] PHQ9_pre > 19: n = 449
## |         (Intercept) TreatmentLow
## |           -1.870597    -1.529347
##
```

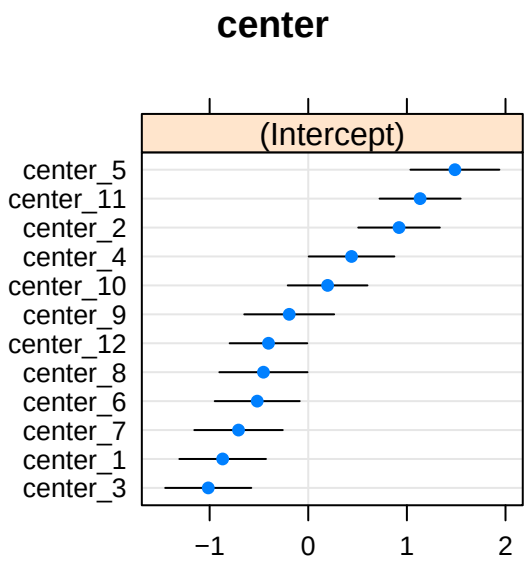
```
## Number of inner nodes:    3
## Number of terminal nodes: 4
## Number of parameters per node: 2
## Objective function (negative log-likelihood): 699.7975
```

```
plot(SMART_t2, which = "tree", gp = gpar(cex = .6))
```




```
plot(SMART_t2, which = "ranef")
```

```
## $center
```



The estimation of the random effects did not yield a different tree structure. But we do see differences between the treatment centers in the probability of recovery; patients in `center_5` have a higher probability of recovery, patients in `center_3` have a lower probability of recovery.

We can request the covariance matrix of the random effects using the `VarCorr` method:

```
VarCorr(SMART_t2)
```

```
##  Groups Name        Std.Dev.
##  center (Intercept) 0.82557
```

9. Detecting Subgroups With Different Reading Trajectories

In this example, we analyse a small subset of the Early Childhood Longitudinal Study-Kindergarten class of 1998-99 (ECLS-K; National Center for Education Statistics, 2016). Data was collected from fall kindergarten 1998 through eighth grade 2007. The goal of the study was to examine child development, school readiness and early school experiences. In this study, we look at assessments of reading skills in kindergarten, first, third, fifth and eighth grade, resulting in five measurement occasions.

The original ECLSK dataset contains data of 21,304 children from schools all across the USA. Here, we analyze a random sample of data from 400 students. We use theta (IRT ability) scores of reading skills, which have a mean of 0 and standard deviation of 1 for the whole sample (but not for the current subsample). The potential partitioning variables are baseline assessments on the child level. These are time-invariant, i.e., not measured on the lowest (observation) level, but on the child (or *cluster*) level. Thus, in addition to estimating a random intercept with respect to an indicator for child, we also have to account for the measurement level of the partitioning variables.

```
ECLSK <- read.table("ECLSK data.txt")
dim(ECLSK)
```

```
## [1] 2000   14
```

```
names(ECLSK)
```

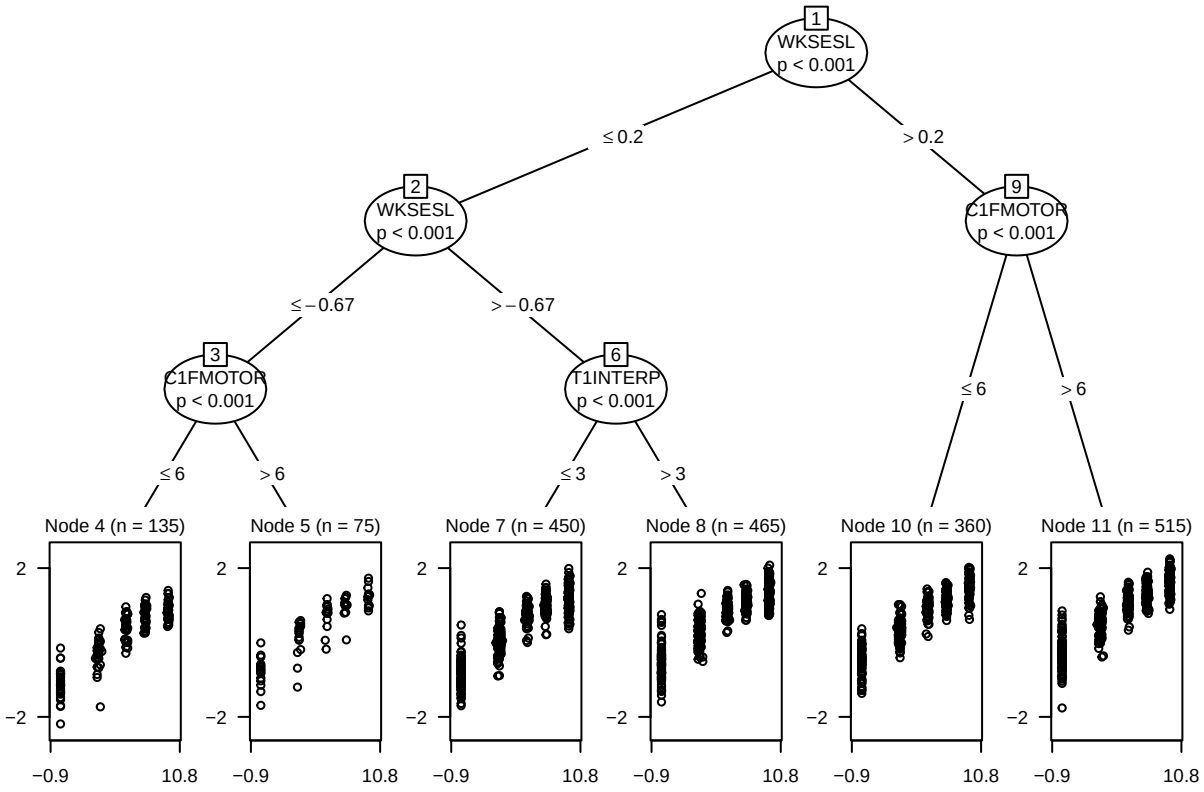
```
##  [1] "GENDER"      "RACE"        "WKSESL"      "C1GMOTOR"    "C1FMOTOR"
##  [6] "T1INTERN"    "T1EXTERN"    "T1INTERP"    "T1CONTRO"    "P1FIRKDG"
## [11] "AGEBASELINE" "score"       "time"        "CHILDID"
```

`score` is the response variable, quantifying reading skills as a theta score from an IRT analysis. `time` is the timing metric and represents the square root of the number of months since baseline. `GENDER`, `RACE` and `AGEBASELINE` are self-explanatory. `WKSESL` is a measure of socio-economic status, `C1GMOTOR` and `C1FMOTOR` are measures of gross and fine motor skills at baseline. `T1INTERN`, `T1EXTERN`, `T1INTERP` and `T1CONTRO` are measures of internalizing, externalizing, interpersonal problems and self control at baseline. `CHILDID` is an indicator for student.

Because `score` is a continuous response variable, we use function `lmertree()` to fit the model. Because the partitioning variables are measured at the cluster level, we specify the `cluster` argument, so that the parameter stability tests will account for the partitioning variables being measured at the child and not the individual observation level:

```
ECLSK_t <- lmertree(score ~ time | CHILDID | GENDER + RACE + WKSESL +
```

```
                      C1GMOTOR + C1FMOTOR + T1INTERN + T1EXTERN +
                      T1INTERP + T1CONTRO + AGEBASELINE,
                    cluster = CHILDID, data = ECLSK)
plot(ECLSK_t, which = "tree", tp_args = list(fitmean = FALSE), gp = gpar(cex = .6))
```



I specified the `fitmean` argument, because for repeated measures data, the plotting the fitted means yields a confusing result. The pattern is difficult to interpret from the plot, but it appears that from left to right, reading scores at every assessment occasion seem to increase somewhat. Thus, higher socio-economic status seems to be associated with higher reading scores. Using the `coef` method, we can obtain the estimated coefficients for every terminal node:

```
coef(ECLSK_t)
```

```
##    (Intercept)      time
## 4   -1.0273482 0.2125477
## 5   -0.6956136 0.2113902
## 7   -0.7277500 0.2168037
## 8   -0.4681510 0.2000110
## 10  -0.4711683 0.2122275
## 11  -0.2576150 0.2011876
```

The estimated coefficients suggest that with higher socio-economic status, higher fine motor skills and higher

interpersonal skills at baseline, children have higher reading skills at baseline, as indicated by the value of the intercepts. Higher intercepts seem associated with lower slopes, so this difference may lessen over time:

```
cor(coef(ECLSK_t))
```

```
##             (Intercept)      time
## (Intercept)    1.000000 -0.678547
## time          -0.678547  1.000000
```

We can obtain the standard deviation of the random intercept term:

```
VarCorr(ECLSK_t)
```

```
##  Groups   Name        Std.Dev.
##  CHILDID  (Intercept) 0.23342
##  Residual             0.23515
```

And we can use that for computing the intraclass correlation coefficient:

```
as.numeric(VarCorr(ECLSK_t)) / var(ECLSK$score)
```

```
## [1] 0.08497081
```

10. Further Reading and Methods

The current workshop aimed to provide a short introduction to recursive partitioning methods (RPMs). The increasing popularity of RPMs, also in the field of machine learning, have resulted in a vast number of RPMs having been developed. Suggestions for further reading and methods are provided below:

Strobl, Malley and Tutz (2009) provide a general but detailed introduction to RPMs aimed specifically at researchers in psychology.

The methodology of conditional inference trees and model-based recursive partitioning is described in Hothorn, Hornik and Zeileis (2006) and Zeileis, Hothorn and Hornik (2008), respectively. The methodology of mixed-effects model trees is described in Fokkema, Smits, Zeileis, Hothorn and Kelderman (2018).

RPMs to detect subgroups in psychometric models (e.g., for the detection of differential item functioning) have been developed by and described in Strobl, Wickelmaier and Zeileis (2011; for recursive partitioning based on Bradley-Terry models); and in Strobl, Kopf and Zeileis (2015; for recursive partitioning based on Rasch models). These methods are implemented in **R** package **psychotree**.

Ensemble methods based on recursive partitioning, where a (large) number of decision trees are combined into one predictive model, have become increasingly popular, especially in the area of machine learning. Two popular tree ensemble methods are random forests (Breiman, 2001; an introduction aimed at researchers in psychology is provided in Strobl, Malley and Tutz, 2009) and boosted tree ensembles (an introduction aimed at researchers in psychology can be found in Miller, Lubke, McArtor and Bergeman, 2016).

Although tree ensemble methods provide state-of-the-art predictive accuracy, the resulting models consist of a large number of trees, which are difficult to interpret. Friedman and Popescu (2008) developed prediction rule ensemble (PRE) methodology, which brings an ensemble of trees back to a smaller ensemble of simple rules, which are easier to interpret while providing very similar predictive accuracy. Fokkema, Smits, Kelderman and Penninx (2015) show how this method can help bridge the gap between clinical and actuarial prediction in psychology. The methodology has been implemented in **R** package **pre** (Fokkema, in press).

11. References

Breiman, L. (2001). Random forests. *Machine Learning, 45*(1), 5-32.

Breiman, L., Friedman, J., Olshen, R., & Stone, C. (1984). *Classification and regression trees.* New York: Wadsworth.

Carrillo, J. M., Rojo, N., Sanchez-Bernardos, M. L., Avia, M. D. (2001). Openness to experience and depression. *European Journal of Psychological Assessment, 17*(2), 130-136.

Edbrooke-Childs, J., Macdougall, A., Hayes, D., Jacob, J., Wolpert, M., & Deighton, J. 9 (2017). Service-level variation, patient-level factors, and treatment outcome in those seen by child mental health services. *European Child & Adolescent Psychiatry, 26*(6), 715-722.

Fokkema, M. (in press). Fitting prediction rule ensembles with **R** package **pre**. *Journal of Statistical Software*. Retrieved from https://arxiv.org/abs/1707.07149

Fokkema, M., Smits, N., Kelderman, H., & Penninx, B. W. (2015). Connecting clinical and actuarial prediction with rule-based methods. *Psychological Assessment, 27*(2), 636.

Fokkema, M., Smits, N., Zeileis, A., Hothorn, T. & Kelderman, H. (2018). Detecting treatment-subgroup interactions in clustered data with generalized linear mixed-effects model trees. *Behavior Research Methods, 50*(5), 2016-2034.

Friedman, J., & Popescu, B. (2008). Predictive learning via rule ensembles. *The Annals of Applied Statistics, 2*(3), 916-954.

Hothorn, T., Hornik, K., Van De Wiel, M. A., & Zeileis, A. (2006). A Lego system for conditional inference. *The American Statistician, 60*(3), 257-263.

Hothorn, T., Hornik, K., Zeileis, A. (2006). Unbiased recursive partitioning: A conditional inference framework. *Journal of Computational and Graphical Statistics, 15*(3), 651-674.

Hothorn, T. & Zeileis, A. (2015). **partykit**: A modular toolkit for recursive partytioning in **R**. *Journal of Machine Learning Research, 16*, 3905-3909.

Loh, W.-Y., & Shih, Y.-S. (1997). Split selection methods for classification trees. *Statistica Sinica, 7*(4), 815-840.

Lucock, M., Barkham, M., Donohoe, G., Kellett, S., McMillan, D., Mullaney, S., ... & Delgadillo, J. (2017). The role of Practice Research Networks (PRN) in the development and implementation of evidence: The Northern improving access to psychological therapies PRN case study. *Administration and Policy in Mental Health and Mental Health Services Research, 44*(6), 919-931.

Miller, P. J., Lubke, G. H., McArtor, D. B., & Bergeman, C. (2016). Finding structure in data using multivariate tree boosting. *Psychological Methods, 21*(4), 583.

National Center for Education Statistics (2016). *Early Childhood Longitudinal Program: Kindergarten class of 1998-1999 (ECLS-K).* Available from National Center of Education Statistics: https://nces.ed.gov/ecls/kindergarten.asp.

Penninx, B. W. J. H., Nolen, W. A., Lamers, F., Zitman, F. G., Smit, J. H., Spinhoven, P., ..., Beekman, A. T. F. (2011). Two-year course of depressive and anxiety disorders: Results from the Netherlands Study of Depression and Anxiety (NESDA). *Journal of Affective Disorders, 133*(1), 76-85.

Strobl, C., Kopf, J., & Zeileis, A. (2015). Rasch trees: A new method for detecting differential item functioning in the Rasch model. *Psychometrika, 80*(2), 289-316.

Strobl, C., Malley, J., & Tutz, G. (2009). An introduction to recursive partitioning: Rationale, application, and characteristics of classification and regression trees, bagging, and random forests. *Psychological Methods, 14*(4), 323.

Strobl, C., Wickelmaier, F., & Zeileis, A. (2011). Accounting for individual differences in Bradley-Terry models by means of recursive partitioning. *Journal of Educational and Behavioral Statistics, 36*(2), 135-153.

Zeileis, A., Hothorn, T., & Hornik, K. (2008). Model-based recursive partitioning. *Journal of Computational and Graphical Statistics, 17*(2), 492-514.