An Introduction to Tree-based Multilevel Models

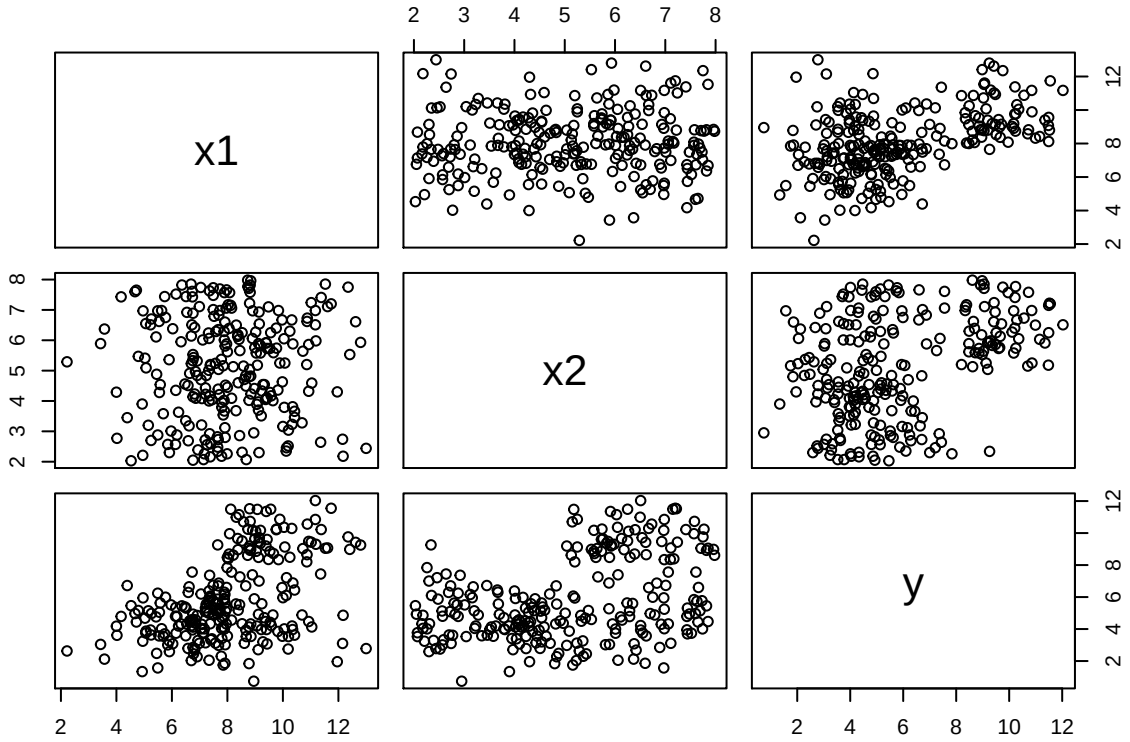
Marjolein Fokkema

20 juni 2018

Trees versus traditional linear models

To illustrate the difference between a traditional GLM and trees, I generated a toy dataset consisting of three continuous variables: $x1$ and $x2$ (two covariates) and y (the response variable). The observed data look as follows:

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



We first take a traditional approach and fit a linear model in which we regress y on $x1$ and $x2$:

```
linear_model <- lm(y ~ x1 + x2, data = toy_data)
summary(linear_model)
```

```
##
## Call:
## lm(formula = y ~ x1 + x2, data = toy_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
```

```
## -5.7997 -1.5850 -0.0328  1.4018  5.2515
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -2.00288    0.69683  -2.874   0.0044 **
## x1           0.61208    0.06961   8.793 2.54e-16 ***
## x2           0.56791    0.08005   7.095 1.37e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.114 on 247 degrees of freedom
## Multiple R-squared:  0.3473, Adjusted R-squared:  0.342
## F-statistic: 65.71 on 2 and 247 DF,  p-value: < 2.2e-16
```

We see that both $x1$ and $x2$ are positively and significantly associated with y. Thus, if we want to predict the values of y for new observations, we could use the following function:

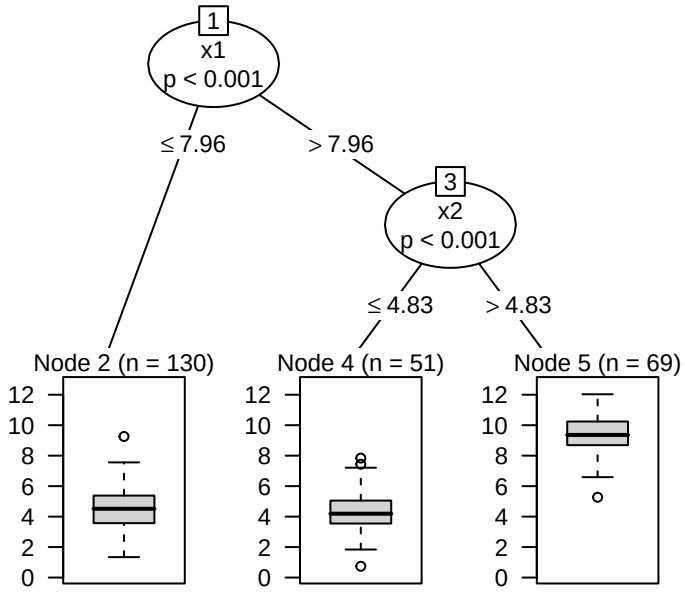
$\hat{y}$ = -2 + 0.61 $x1$+ 0.57 $x2$

But who likes formulas anyway? (Well, I do, but I'm a statistician.)

One of the major downsides of this formula is that it is not immediately clear for what values of $x1$ and $x2$ we would expect strikingly higher or lower than average values of y. For example, if y was a depression index, which persons would be at higher (lower) than average risk of having depression?

Fitting a tree may provide this insight, so we fit one using the partykit package:

```
library("partykit")
lm_tree <- lmtree(y ~ 1 | x1 + x2, data = toy_data)
plot(lm_tree, gp = gpar(cex = .75))
```

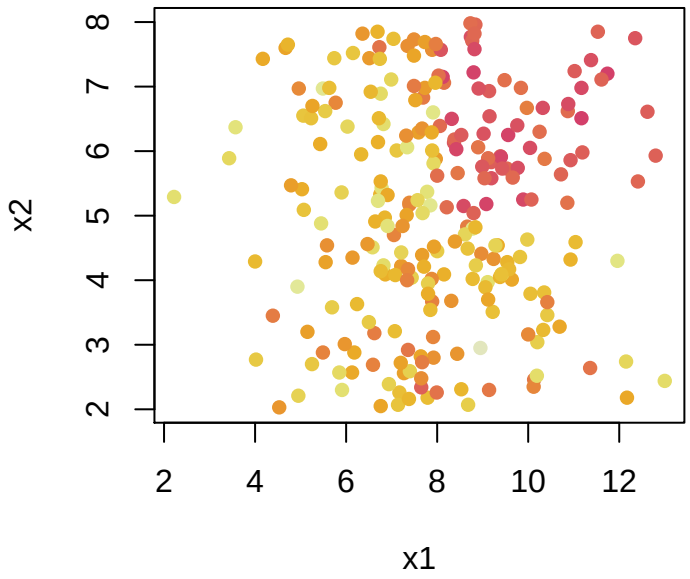


The first argument supplied to the `lmtree()` function specifies the formula. We specify the node-specific model by regressing the response variable on an intercept (`bdi ~1`). We specify the possible partitioning (`x1 + x2`) variables after the vertical bar (`|`). In words, this could be read as: 'Regress y on an intercept, conditional on $x1$ and $x2$'.

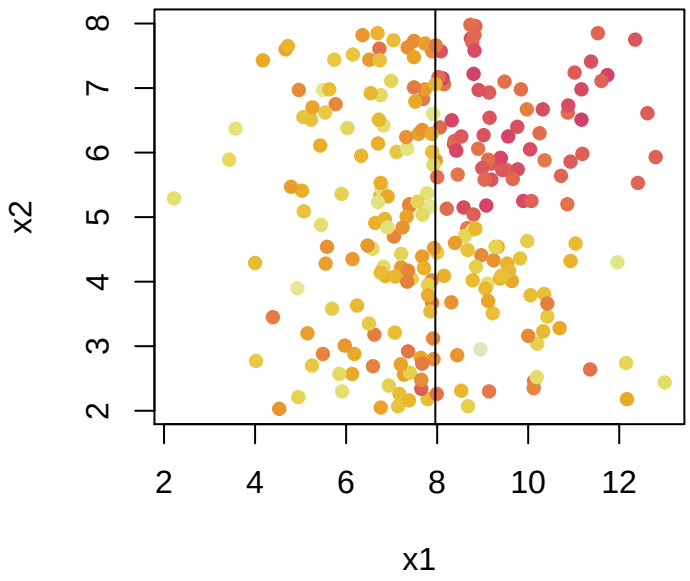
The tree reveals three subgroups: a subgroup with lower values of $x1$, which shows lower values for y (node 2); a subgroup with higher values of $x1$ and lower values of $x2$, which also shows lower values for y (node 4); and a subgroup with higher values values of $x1$ and higher values of $x2$ and markedly higher values of y (node 5). Note that the tree indicates an interaction effect (the effect of $x2$ is dependent on the value of $x1$), which was not reflected in the linear model.

Every split in the tree shows a p-value. This is the p-value for the parameter stability test, which guides the tree construction process. To select the splitting variable in each node, a parameter stability test is performed. The covariate (or partitioning variable) yielding the lowest p-value is then selected for splitting in the current node. When none of the partitioning variables yield a p-value below the pre-specified α level ($\alpha = .05$ by default), the splitting process in that node is stopped. An extensive discussion of the splitting procedure is outside of the scope of this paper, but a short popular explanation will follow below. A short technical explanation is provided in the Appendix.

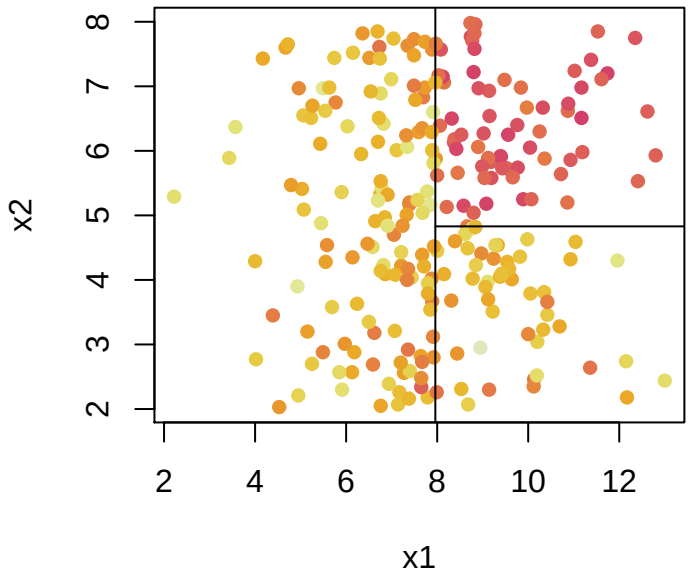
The tree above provides a graphical representation of a recursive partition: at each split, the observations are separated into increasingly similar subgroups with respect to the outcome variable y. As we have only two covariates in this example, we could depict this partitioning process as follows:



We start with the complete dataset in the root node. Then we try to find a splitting variable and value which would yield two subgroups as similar as possible with respect to y. This variable is $x1$, with a value of 7.96:



In the left subgroup (panel), the remaining possible splitting variables and values would not yield subgroups that are more similar than the current one. In the right subgroup (panel), we continue splitting as the upper part shows higher values of y than the lower part of the right panel. Thus, splitting by $x2$ with a value of would yield two subgroups which are as similar as possible with respect to y:



And then we stop splitting altogether, because the remaining possible splitting variables and values would not yield subgroups that are more similar then the current ones.

We will now go to the first applied example, in which we fit a tree for predicting depression, based on personality scales.

Example 1: Predicting depression

For this example, we will make use of the dataset from a study by Carrillo et al. (2001), who examined the association between big five personality characteristics (as measured by NEO-PI-R (sub)scale scores), sex and age, and depression (as measured by the Beck Depression Inventory). It is available as file "carrillo data.txt".

```
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"
```

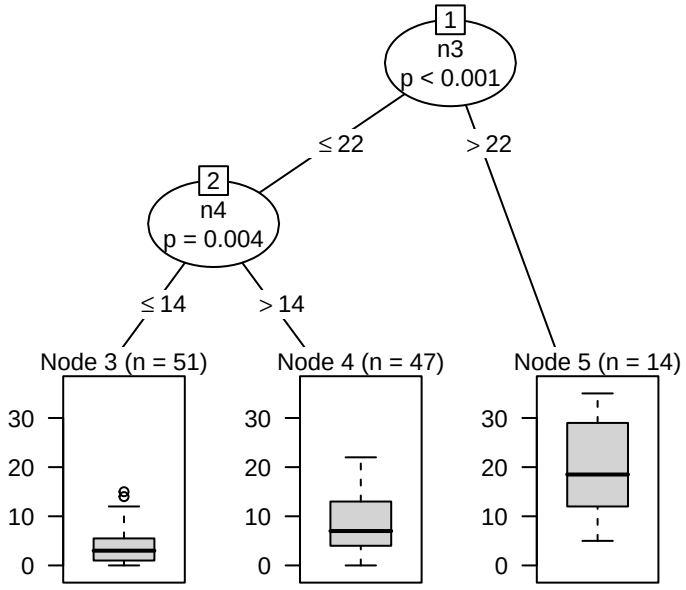
```
dim(carrillo)
```

```
## [1] 112  26
```

```
car_tree <- lmtree(bdi ~ 1 | ., data = carrillo)
```

Note that, because there are so many possible partitioning variables, we use the dot after the vertical bar (`.`) as a short-hand notation to specify that all variables that were not explicitly mentioned in the formula, should be used as possible partitioning variables.

```
plot(car_tree, gp = gpar(cex = .75))
```



The plot reveals that two covariates were most predictive of BDI scores: $n3$ and $n4$ (two neuroticism facet scales). Thus, other personality characteristics, sex and age do not seem to add any predictive value over and above the two neuroticism facet scores. Also, the p-values shown in the plot are Bonferroni corrected

for multiple testing. This could have been turned off by adding `bonferroni = FALSE` to the `lmtree()` specification, but this would have substantially increased the likelihood of Type-I errors. Also, the α value (the a-priori specified probability of incorrectly rejecting the null hypothesis of parameter stability) can be adjusted by adding `alpha = .10`. The default employed by `lmtree()` is $\alpha = .05$.

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

```
car_tree
```

```
## Linear model tree
##
## Model formula:
## bdi ~ 1 | .
##
## Fitted party:
## [1] root
## |   [2] n3 <= 22
## |   |   [3] n4 <= 14: n = 51
## |   |       (Intercept)
## |   |           3.72549
## |   |   [4] n4 > 14: n = 47
## |   |       (Intercept)
## |   |          8.808511
## |   [5] n3 > 22: n = 14
## |       (Intercept)
## |          20.64286
##
## Number of inner nodes:    2
## Number of terminal nodes: 3
## Number of parameters per node: 1
## Objective function (residual sum of squares): 3622.648
```

Trees for Multilevel and Longitudinal Datasets

Above, we assumed the observations were independently sampled from the population. In many studies, data is collected with some dependence structure. For example, patients may be sampled within geographical areas, hospitals and/or physicians, or children may be sampled within classrooms and/or schools. This introduces dependence between observations: Observations from the same higher-level unit are more alike than observations from different higher-sampling units. One way to account for this structure by estimation of random effects. We can do this by fitting generalized linear mixed-effects regression trees with the `glmertree` package:

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

The two main functions of the `glmertree` package are `lmertree()`, which can be used for continuous response variables and `glmertree()`, which can be used for binary and count responses. We will illustrate their use through an example:

Example 2: Predicting treatment outcomes in a multi-center dataset

For this exmaple, we will make use of an artificial dataset modelled after Edbrooke-Childs et al. (2017), who analyzed a sample of $N = 3,739$ young people who received treatment at one of 13 mental-health service providers in the UK. Potential predictor variables were demographic variables (age, gender, ethnicity), case characteristics (coding the presence or absence of several mental and behavioral disorders) and severity

characteristics (measures of impairment in functioning) assessed at baseline. The response variable was treatment outcome, as measured by a total mental-health difficulties score assessed about 6 months after baseline, corrected for the baseline assessment. Thus, higher values indicate a poorer treatment result or outcome.

Note that the data were artificially generated and the results are not valid in the real world.

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

```
## [1] 3739    8
```

```
head(UKMH_data)
```

```
##    age impact gender emotional autism conduct cluster_id outcome
## 1 16.0    4.9 female       yes     no      no          8    -0.2
## 2  9.4    4.4 female       yes     no      no          1     0.2
## 3 12.6    2.5   male        no     no      no          2    -0.6
## 4 13.5    3.7   male       yes     no     yes         13     0.0
## 5 12.7    0.9 female       yes     no      no         12     0.7
## 6 11.0    3.6   male        no     no      no          7     0.0
```

Thus, the dataset includes two continuous covariates (age and impact); four binary covariates (gender, presence of emotional problems, autism and conduct problems), an indicator for service provider (cluster_id) and a continuous variable reflecting treatment outcome (outcome). We will estimate a mixed-effects regression tree with the covariates as potential partitioning variables and a random intercept with respect to the cluster_id variable:

```
UKMH_tree <- lmertree(outcome ~ 1 | cluster_id | age + gender + emotional +
                        autism + impact + conduct, data = UKMH_data)
```

Note that, compared to the first example, we have to additionally specify the random-effects structure. The formula statement consists of three parts: The first part specifies the node-specific model (`outcome ~ 1`); the second part (after the first vertical bar) specifies the random effects structure (`cluster_id`); the third part (after the second vertical bar) specifies the potential partitioning variables (`age + gender + emotional + autism + impact + conduct`). Note that due to the more complex modeling structure of mixed-effects trees, use of the `.` to specify all remaining variables at once is not advised (the wrong variables may end up in the wrong part of the model). Also note that we employ a very simple random-effects specification here. That is, we only estimate a random intercept with respect to the `cluster_id` variable. Therefore, we can use the short-hand notation `| cluster_id |`, specification of `| (1 | cluster_id) |` would have yielded the exact same model. More complex random-effects structures can also be specified:

Specify random intercepts with respect to multiple cluster indicators:

For example, if our dataset would contain the variables `service` and `department`, where patients are nested within departments, which are nested within services (i.e., nested design):

```
outcome ~ 1 | ( (1 | service / department) ) | age + gender +
  emotional + autism + impact + conduct
```

If we would have a crossed design, where there are multiple cluster indicators (e.g., `cluster_id1` and `cluster_id2`), but the units of one indicator are not nested within the units of the other (i.e., crossed design):

```
outcome ~ 1 | ( (1 | cluster_id1) + (1 | cluster_id2) ) | age + gender +
  emotional + autism + impact + conduct
```

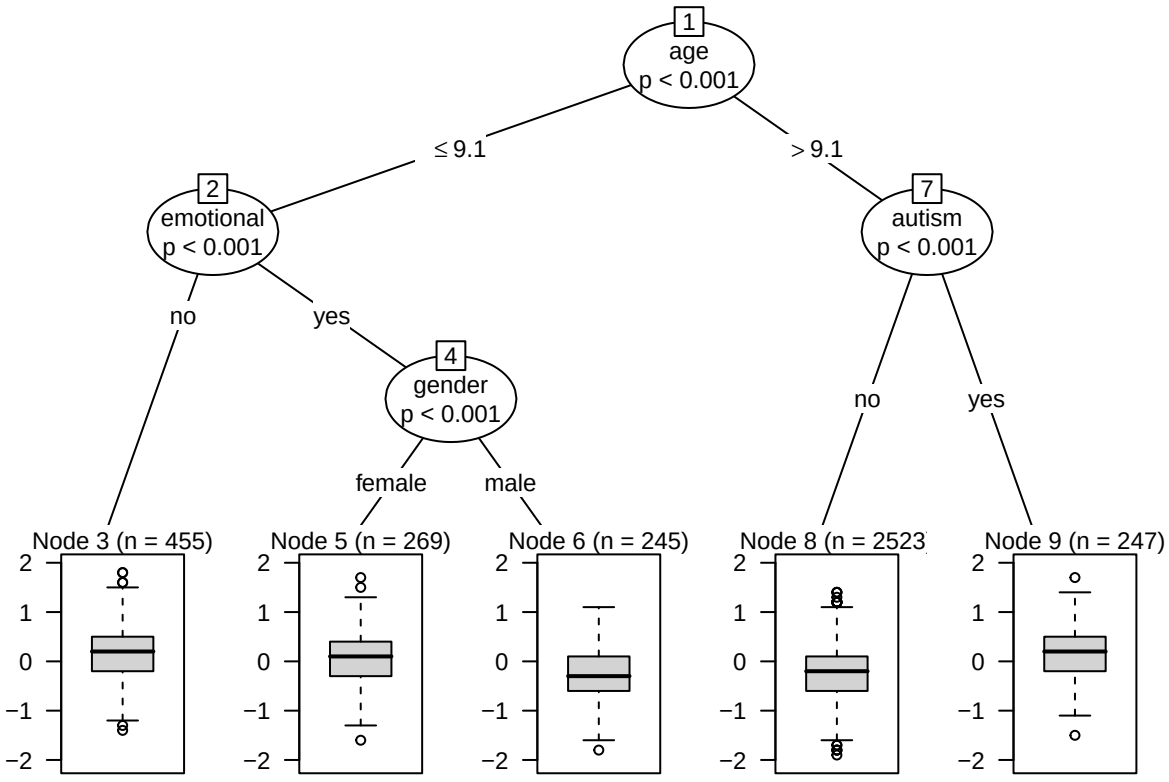
Specify random slopes in addition to random intercepts:

If `outcome` were the treatment outcome and we also have a variable `baseline` for which we want to correct, but we expect the effect of the baseline measurement on the outcome to differ over clusters (level-2 units), we could specify a random slope in addition to the random intercept for `cluster_id`:

```
outcome ~ 1 | (1 + baseline | cluster_id) | age + gender +
  emotional + autism + impact + conduct
```

Inspecting the fitted mixed-effects tree

```
plot(UKMH_tree, which = "tree", gp = gpar(cex = .75))
```




```
UKMH_tree$tree
```

```
## Linear model tree
##
## Model formula:
## outcome ~ 1 | age + gender + emotional + autism + impact + conduct
##
## Fitted party:
## [1] root
## |   [2] age <= 9.1
## |   |   [3] emotional in no: n = 455
## |   |       (Intercept)
## |   |         0.1870159
## |   |   [4] emotional in yes
## |   |   |   [5] gender in female: n = 269
## |   |   |       (Intercept)
```

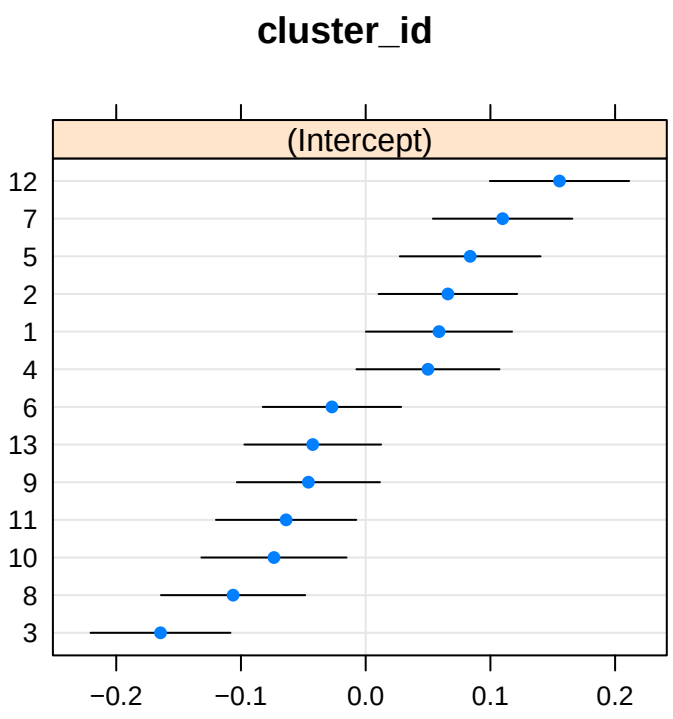
```
## |   |   |         0.1044675
## |   |   |   [6] gender in male: n = 245
## |   |   |       (Intercept)
## |   |   |        -0.2731195
## |   [7] age > 9.1
## |   |   [8] autism in no: n = 2523
## |   |       (Intercept)
## |   |        -0.2252742
## |   |   [9] autism in yes: n = 247
## |   |       (Intercept)
## |   |         0.1804013
##
## Number of inner nodes:    4
## Number of terminal nodes: 5
## Number of parameters per node: 1
## Objective function (residual sum of squares): 975.3689
```

The fitted tree identified four variables predictive of treatment outcome: age, presence of emotional problems, autism, and gender. The best treatment outcomes (i.e., lowest values on the response variable) are observed among males presenting with emotional problems that entered mental health services at age ≤ 9.1. The poorest treatment outcomes (i..e, highest values on the response) are observed among those that entered mental health services at age ≤ 9.1 and did not present with emotional problems.

In addition, we should inspect the estimated random effects:

```
plot(UKMH_tree, which = "ranef", width = 0.5)
```

```
## $cluster_id
```



Or through printing:

```
ranef(UKMH_tree)
```

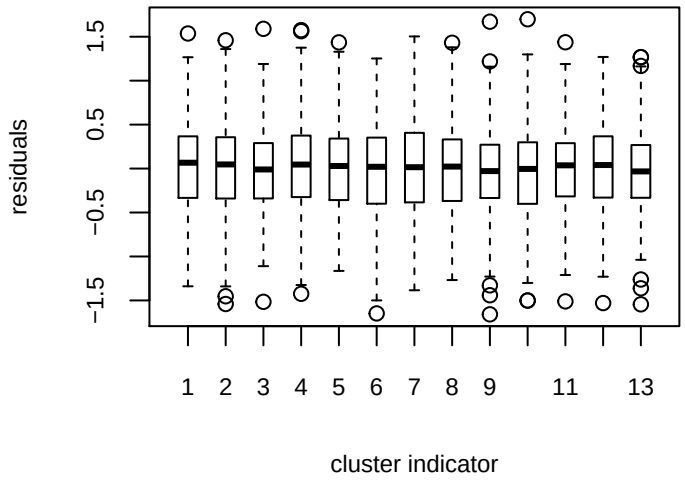
```
## $cluster_id
##    (Intercept)
## 1   0.05878157
## 2   0.06587671
## 3  -0.16449595
## 4   0.04992008
## 5   0.08378504
## 6  -0.02704807
## 7   0.10980018
## 8  -0.10630876
## 9  -0.04592325
## 10 -0.07353663
## 11 -0.06379060
## 12  0.15539792
## 13 -0.04245822
```

Checking for model (mis-)specification

One of the advantages of tree-based methods is that they involve little assumptions about the data distribution. However, one should always check whether the results of the data analysis are in line with the specified model, especially for mixed-effects models. A large part of checking can be done by plotting the tree and random effects: This provides an overview of differences between observations within and between nodes, possible outliers, and clusters with markedly higher or lower values on the response variable.

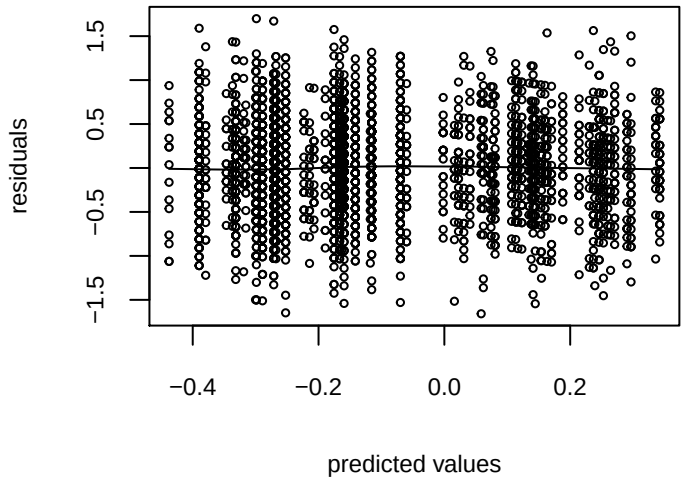
Furthermore, we should check for heteroscedasticity. Firstly, by assessing whether the model's residuals systematically differ as a function of cluster membership:

```
resids <- residuals(UKMH_tree)
plot(x = factor(UKMH_data$cluster_id), y = resids,
     xlab = "cluster indicator", ylab = "residuals",
     boxwex = .5, cex.lab=.75, cex.axis = .75)
```



Or as a function of the model's predictions:

```
preds <- predict(UKMH_tree)
scatter.smooth(x = preds, y = resids, xlab = "predicted values",
               ylab = "residuals", cex = .5, cex.lab = .75, cex.axis = .75)
```



In this example, the plots indicate no systematic pattern in the residuals with respect to cluster membership or predicted values, which is good.

Example 3: Detecting treatment outcome predictors and moderators in multilevel data

For this example, we will make use of an artificial dataset inspired by the SMART tournament data. The artificial dataset is available in the file "SMART mimic data.txt". Although the dataset is here referred to as `SMART_data`, it is not the actual SMART data and the results below are therefore not valid in the real world.

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

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

```
names(SMART_data)
```

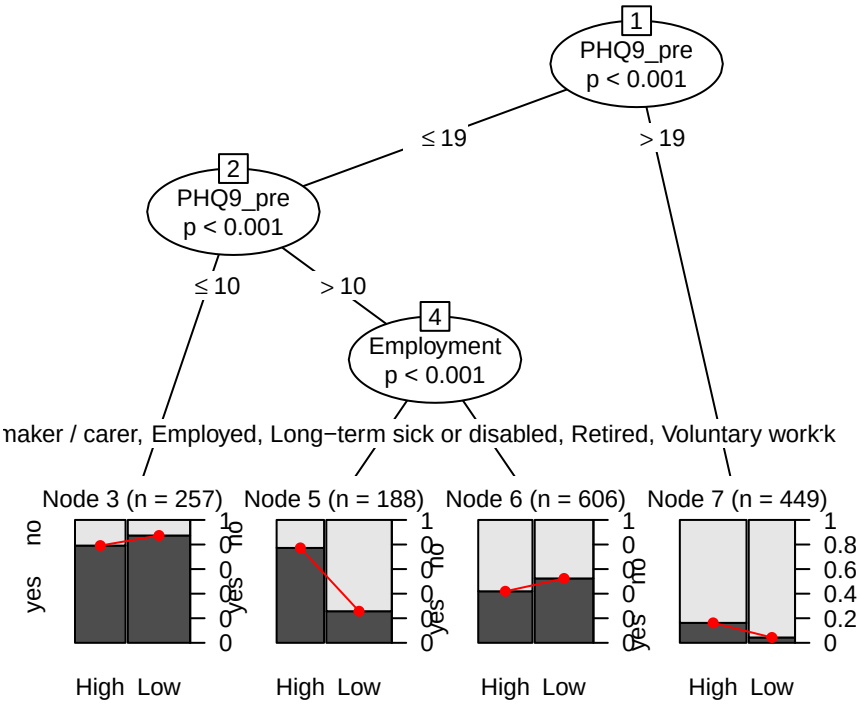
```
##  [1] "recovered"  "Treatment"  "Age"        "Gender"     "Ethnicity"
##  [6] "Diagnosis"  "Employment" "Disability" "PHQ9_pre"   "GAD7_pre"
## [11] "WSAS_pre"   "Medication" "center"
```

The dataset consists of 1,500 observations and 13 variables: A binary response variable (`recovered`), an indicator for treatment type (`Treatment`), four continuous partitioning covariates (`Age`, `PHQ9_pre`, `GAD7_pre` and `WSAS_pre`), six categorical partitioning covariates (`Gender`, `Ethnicity`, `Diagnosis`, `Employment`, `Disability` and `Medication`) and an indicator for treatment center (`center`).

Here, we fit a tree to detect predictors of recovery and moderators of treatment effect among the partitioning covariates. We therefore regress the response variable on the treatment indicator (i.e., `recovered ~ Treatment`). The `Treatment` variable indicates whether a patient received low or high intensity treatment. In addition, we account for differences between treatment centers by estimating a random intercept with respect to `center`. Because the response variable is binary, we have to employ the `glmertree()` function, and have to specify `family = binomial`:

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

```
plot(SMART_tree, which = "tree", gp = gpar(cex = .75))
```




```
SMART_tree
```

```
## Generalized linear mixed model tree
##
## 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
## |   |           1.5288250    0.5570944
## |   |   [4] PHQ9_pre > 10
## |   |   |   [5] Employment in Homemaker / carer, Student, Unemployed job seeker, Unemployed, not seek
## |   |   |         (Intercept) TreatmentLow
## |   |   |            1.208380    -2.467001
## |   |   |   [6] Employment in Employed, Long-term sick or disabled, Retired, Voluntary work: n = 606
## |   |   |         (Intercept) TreatmentLow
## |   |   |          -0.3028141    0.4358695
## |   [7] PHQ9_pre > 19: n = 449
## |         (Intercept) TreatmentLow
## |           -1.873063    -1.530169
##
## Number of inner nodes:    3
## Number of terminal nodes: 4
## Number of parameters per node: 2
```

```
## Objective function (negative log-likelihood): 698.7105
##
## Random effects:
## $center
##           (Intercept)
## center_1   -0.8718304
## center_10   0.1915106
## center_11   1.1328605
## center_12  -0.4212284
## center_2    0.9205330
## center_3   -1.0200065
## center_4    0.4797589
## center_5    1.4964775
## center_6   -0.5252440
## center_7   -0.7096820
## center_8   -0.4583912
## center_9   -0.1946419
```

Note that the width of the bars reflect the number of observations in that node, within the specific treatment condition. Thus, the left-most terminal node (node 3) contains somewhat more observations in the low intensity than in the high intensity treatment condition. Also note that the rather long labels for the levels of the Employment variable make the plotted tree a bit more difficult to interpret, so we also printed the tree.

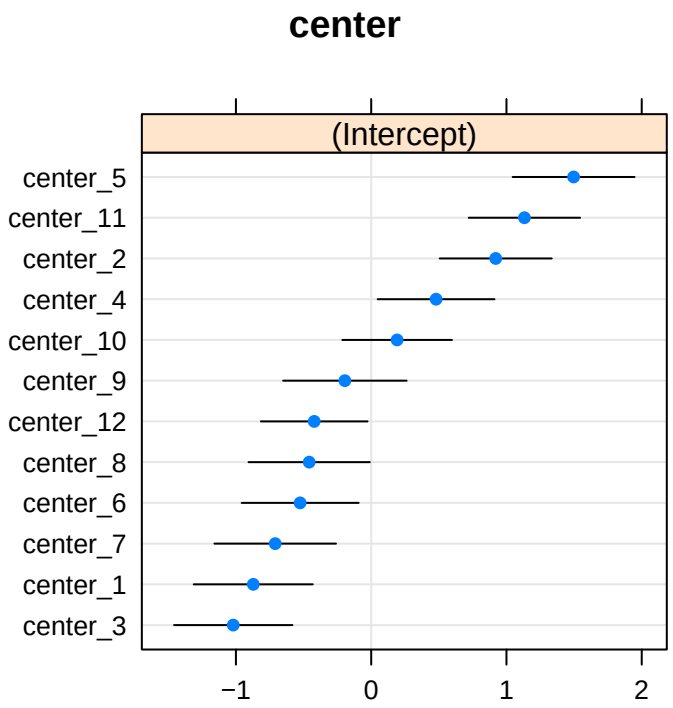
The left-most node also shows that patients with PHQ9 scores ≤ 9 at the start of treatment are likely to recover over the course of treatment, and even seem somewhat more likely to recover with low intensity treatment than high intensity treatment.

Furthermore, for patients with PHQ9 scores > 10 but ≤ 19, employment status is an important predictor of treatment efficacy: For homemaker/carers, students, unemployed job seekers and non-seekers, high intensity treatment seems much more effective. For those patients who are employed, long-term sick or disabled, retired, or voluntary workers, the low intensity treatment seems somewhat more effective, but the difference between the two treatment conditions is small.

Finally, for patients with strikingly high PHQ9 values at the start of treatment, recovery is unlikely, but high intensity treatment may be slightly more effective than low intensity treatment.

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

```
## $center
```



The plot of the estimated random effects shows that `center_5` has a relatively high and `center_3` has a relatively low value for the intercept. This indicates that on average, the likelihood of recovery in `center_5` is higher than in `center_3`.

References

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., Smits, N., Zeileis, A., Hothorn, T. & Kelderman, H. (in press). Detecting treatment-subgroup interactions in clustered data with generalized linear mixed-effects model trees. Behavior Research Methods.

Zeileis, A. & Fokkema, M. (2017). glmertree: Generalized Linear Mixed Model Trees. R package version 0.1. url: https://cran.r-project.org/package=glmertree.

Hothorn, T., Seibold, H. & Zeileis, A. (2018). partykit: A Toolkit for Recursive Partytioning. R package version 1.2-2. url: https://cran.r-project.org/package=partykit.

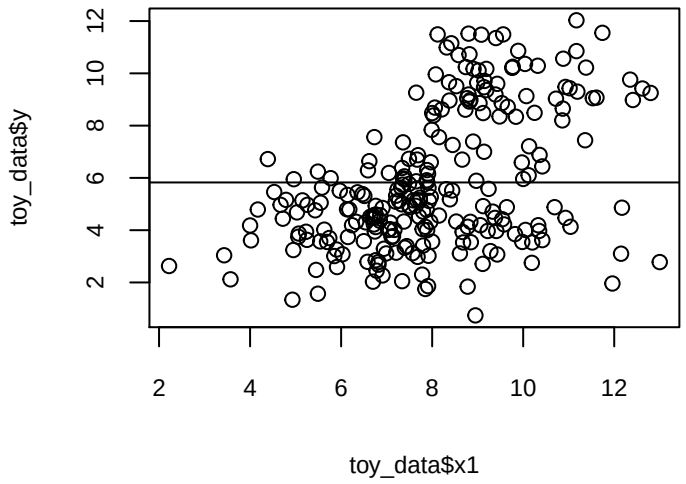
Zeileis, A., Hothorn, T., & Hornik, K. (2008). Model-based recursive partitioning. Journal of Computational and Graphical Statistics, 17 (2), 492-514.

Appendix: Parameter stability tests

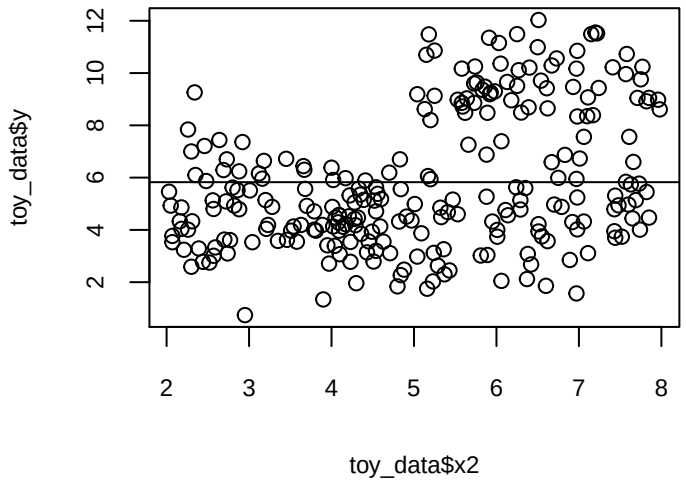
Splitting variables are selected based on so-called parameter stability tests. These parameter stability tests capture fluctuations of a model's scores. The model in the tree in the `toy_data` example is an intercept only

model (that is, the first part of the formula statement is `y ~ 1`). Thus, our null model consists of a single mean. Let us take a look at how the observed values differ from the estimated model:

```
mean_y <- mean(toy_data$y)
plot(x = toy_data$x1, y = toy_data$y, cex.lab = .75, cex.axis = .75)
abline(mean_y, 0)
```




```
plot(x = toy_data$x2, y = toy_data$y, cex.lab = .75, cex.axis = .75)
abline(mean_y, 0)
```



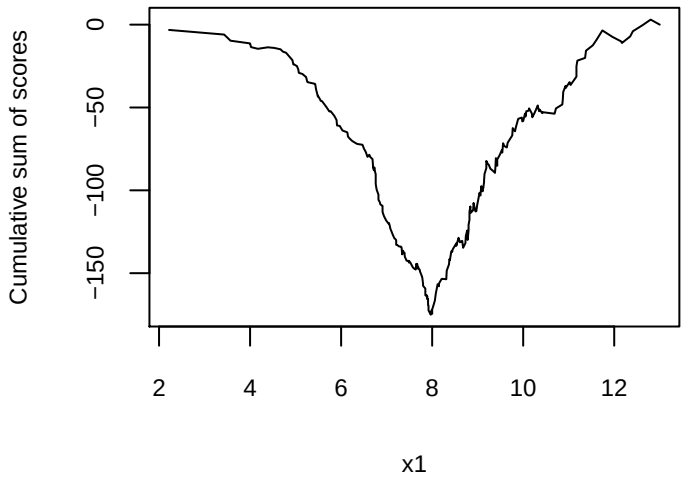
The plots indicate that our 'one-mean-fits-all' hypothesis is probably incorrect: higher values of $x1$ show higher deviations from the mean, the same goes for $x2$. To quantify these deviations from the null model, we

use the model's scores. In our model, with a continuous response and covariates, and an intercept only (i.e., no slopes), the scores happen to be equal to the residuals (i.e., observed minus predicted y):

```
scores <- toy_data$y - mean_y
```

The expected value (mean) of the scores is 0, by definition. Therefore, the scores always sum to 0. If we order the scores by the values of a covariate, any systematic deviation in the scores from their expected value of zero would indicate dependence between the covariate and the response. Systematic deviations can be captured through calculating the cumulative sums:

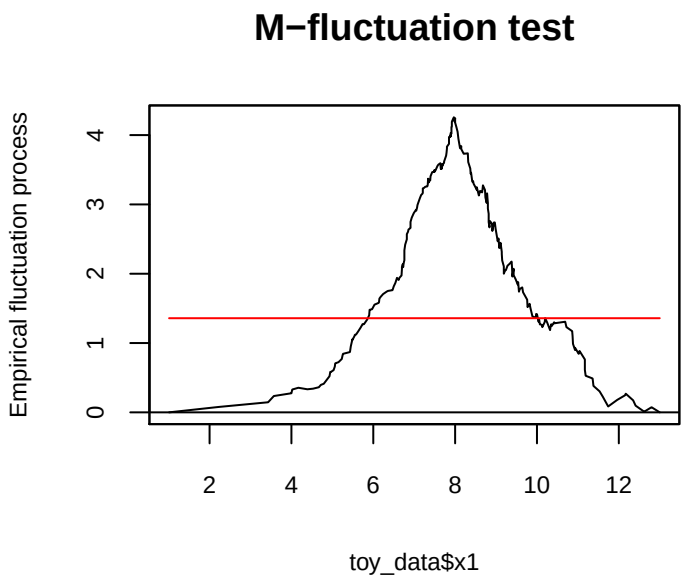
```
index <- order(toy_data$x1)
cs_scores <- cumsum(scores[index])
plot(toy_data$x1[index], cs_scores, type = "l", xlab = "x1",
     ylab = "Cumulative sum of scores", cex.lab = .75, cex.axis = .75)
```



This plot indicates the scores systematically deviate from their mean of 0, with respect to covariate $x1$: starting with $x1 = 2$, the scores are negative (summing up to an increasingly lower value), while from about $x1 = 8$, the scores show positive values, summing up to increasingly higher values. This is also what we saw in the plotted data points above: for lower values of $x1$, most points were below the mean (i.e., negative residuals) and for higher values of $x1$, most points were above the mean (i.e., positive residuals). This indicates that there are probably two subgroups with different means of the response variables, which would best be separated by $x1$ with a value of about 8. That is why the tree selected that splitting variable and value first.

Note that this is a very simplified version of what the `lmtree()` function actually computes. The computational details of splitting variable and value selection are provided in Zeileis, Hothorn & Hornik (2008). In fact, the `lmtree()` and `glmtree()` (and also the `lmertree()` and `glmertree()` functions) uses function `gefp()` (short for generalized empirical fluctuation process) from package strucchange to calculate the p-values:

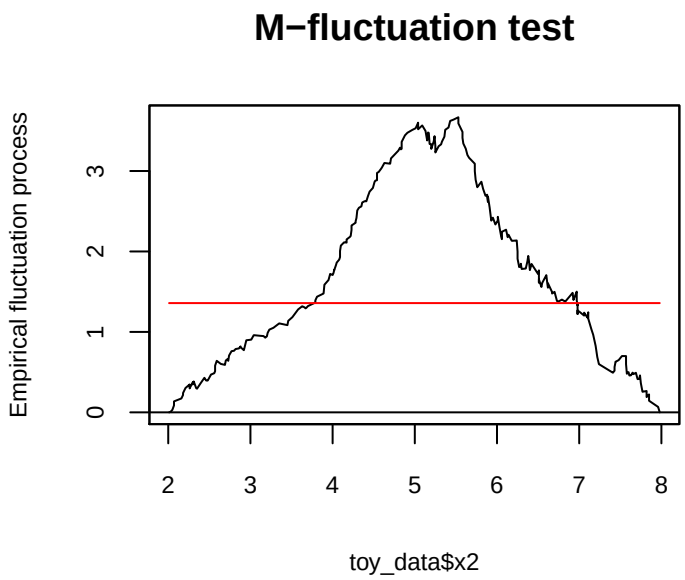
```
library("strucchange")
gefp_x1 <- gefp(lm(y ~ 1, data = toy_data), fit = NULL, order.by = toy_data$x1)
plot(gefp_x1, cex.lab = .75, cex.axis = .75)
```



Note that the plot for $x1$ created with `gefp()` has the exact same shape as the plotted cumulative sums above. Only the scaling is different: `gefp()` takes the absolute values, and uses sample size and a (co)variance matrix to scale the values. As a result, the empirical fluctuation process values calculated by `gefp()` can be used as test statistics, which can be compared with a critical value based on an a-priori specified α level (for falsely rejecting the null hypothesis of parameter stability). That critical value is depicted by the red horizontal line in the plot above. If we would have specified a different value (e.g., by adding `alpha = .25`), the red line would have been lower and a statistically significant result would have been obtained earlier.

We can request the same plot for $x2$. This yields a very similar result as for $x1$, but because $x1$ has a slightly higher peak than $x2$, $x1$ was selected for the first split.

```
gefp_x2 <- gefp(lm(y ~ 1, data = toy_data), fit = NULL, order.by = toy_data$x2)
plot(gefp_x2, cex.lab = .75, cex.axis = .75)
```


M−fluctuation test
Empirical fluctuation process
0
1
2
3
2
3
4
5
6
7
8
toy_data$x2