Fitting Generalized Linear Mixed-Effects Model Trees

Marjolein Fokkema
Universiteit Leiden

Achim Zeileis
Universität Innsbruck

Abstract

This vignette introduces the **glmertree** package for fitting generalized linear mixed-effects model trees (GLMM trees or glmertrees). In hands-on examples based on artificial datasets, emphasis is given to three special cases of fitting GLMM trees: trees with constant fits in the terminal nodes (Section 2), detection of treatment-subgroup interactions (Section 3), and subgroup detection in longitudinal growth curve models (Section 4).

Keywords: recursive partitioning, mixed-effects model trees, decision trees.

1. Introduction

Generalized linear mixed-effects model trees (GLMM trees or glmertrees) have initially been proposed by Fokkema, Smits, Zeileis, Hothorn, and Kelderman (2018) for detecting treatment-subgroup interactions in clustered datasets. However, GLMM trees may be used to address a wider range of research questions. Using hands-on examples based on artificial datasets, this vignette describes how to fit GLMM trees. Section 2 shows how to fit (G)LMM trees with constant fits in the terminal nodes on clustered (multilevel) data. Section 3 shows how to assess main and interaction effects of a categorical variable (treatment) on a continuous response (treatment outcome) on clustered data. Section 4 shows how subgroups can be detected with respect to the parameters of a growth curve model. These are just examples; package **glmertree** can be used to detect predictors and moderators in a wide range of GLMMs-type models.

The GLMM tree model is composed of global and local parts: The global model is composed of the random-effects terms and using all observations. The local model is composed of the fixed-effects terms, which are estimated locally: the observations in a dataset are partitoned with respect to additional covariates (a.k.a. partitioning variables) and a separate fixed-effects model is estimated in each cell of the resulting partition. Package **glmertree** employs package **partykit** (Hothorn and Zeileis 2015) to find the partition and package **lme4** (Bates, Mächler, Bolker, and Walker 2015) to estimate the mixed-effects model.

The current stable release version of package **glmertree** can be installed from the Comprehensive R Archive Network (CRAN) as follows:

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

Alternatively, the current development version can be installed from R-Forge:

```
R> install.packages("glmertree", repos = "http://R-Forge.R-project.org")
```

After installation, the package can be loaded as follows:

```
R> library("glmertree")
```

The main functions in package **glmertree** are `lmertree()` for continuous outcome variables, and `glmertree()` for binary or count outcome variables. Both functions require specification of a `formula` and `data` argument, as will also be illustrated in the examples below. The various methods that are available to plot and print the fitted models and to extract further information will also be shown in the examples below.

For both main functions, the `formula` argument specifies the model formula, which is composed of a left- and right-hand side: the left-hand side specifies the response variable, followed by a tilde (`~`). The right-hand side comprises three parts: the predictors for the node-specific model (comprising fixed effects only, with coefficients that are allowed to differ over subgroups), the global model (comprising random and/or fixed effects, for which coefficients are estimated globally, using all observations) and the potential partitioning variables. The three parts of the right-hand side are separated by vertical bars:

```
response ~ node-specific predictors | global predictors | partitioning variables
```

2. Fitting a mixed-model tree with constant fits

For this example, we will make use of the artificially generated `MHserviceDemo` dataset, containing data on $N = 350$ young people receiving treatment at one of 13 mental-health service providers. The response variable is (`outcome`), a continuous variable representing treatment outcome, as measured by a mental-health difficulties score at follow-up, corrected for the baseline assessment, with higher values reflecting poorer treatment outcome. Potential predictor variables are demographic variables and case characteristics: two continuous (`age` and `impact`) and four binary covariates (`gender`, `emotional`, `autism` and `conduct`). The cluster indicator (`cluster_id`) is an indicator for the mental-health service provider. The data can be loaded as follows:

```
R> data("MHserviceDemo", package = "glmertree")
R> summary(MHserviceDemo)
```

```
      age             impact          gender    emotional autism   
 Min.   : 1.100   Min.   :-5.600   female:162   no :153   no :317  
 1st Qu.: 9.025   1st Qu.: 2.000   male  :188   yes:197   yes: 33  
 Median :11.250   Median : 4.500                                   
 Mean   :11.233   Mean   : 4.229                                   
 3rd Qu.:13.500   3rd Qu.: 6.275                                   
 Max.   :20.600   Max.   :14.200                                   
                                                                   
 conduct     cluster_id     outcome       
 no :285   13     : 35   Min.   :-1.8000  
 yes: 65   4      : 33   1st Qu.:-0.5000  
           11     : 33   Median :-0.2000  
 12      : 32   Mean   :-0.1406
 2       : 31   3rd Qu.: 0.2000
 8       : 31   Max.   : 1.6000
 (Other):155
```

The response is a continuous variable, so we employ function `lmertree()`. In the model formula, we specify `outcome` as the response variable, followed by a tilde. Next, we specify the node-specific model, which comprises only an intercept in this case, because we want to identify subgroups which differ on their value of the response variable. Next, we specify the predictor for the global model. We only want to account for possible outcome differences between the service providers, so we specify a random intercept with respect to the `cluster_id` variable. Finally, we specify the demographic variables and case characteristic as the potential partitioning variables:

```
R> MH_tree <- lmertree(outcome ~ 1 | cluster_id | age + gender + emotional +
+                         autism + impact + conduct, data = MHserviceDemo)
```

Note that we specified `cluster_id` as a predictor variable for the global model, in order to estimate arandom intercept with respect to `cluster_id`. We used short-hand notation for `(1|cluster_id)`; because function `lmertree` and `glmertree` assume a single random intercept term, by default, which is specified through providing the cluster indicator only.

More complex random-effects structures can be specified with the customary formulation employed in package **lme4**. For example, if we would want to account for a global linear fixed effect of age, we can incorporate it in the specification of the global model as follows (results not presented or discussed further):

```
R> MH_tree2 <- lmertree(outcome ~ 1 | age + (1 | cluster_id) | gender +
+                          emotional + autism + impact + conduct,
+                        data = MHserviceDemo)
```

Note that we used the round brackets around `(1|cluster_id)` to protect the vertical bars separating the global predictors from the node-specific predictors and potential partitioning variables.

Alternatively, using the `glmertree()` function, a tree may be fitted to binary (`family = binomial`, default) or count response variables (`family = poisson`). Therefore, a binomial GLMM tree for a dichotomized response could be obained by (results not presented or discussed further):

```
R> MHserviceDemo$outcome_bin <- factor(MHserviceDemo$outcome > 0)
R> MH_gtree <- glmertree(outcome_bin ~ 1 | cluster_id | age + gender +
+                           emotional + autism + impact + conduct,
+                         data = MHserviceDemo, family = "binomial")
```

Using the `plot` method, we can plot the resulting tree and random effects:

```
R> plot(MH_tree)
```

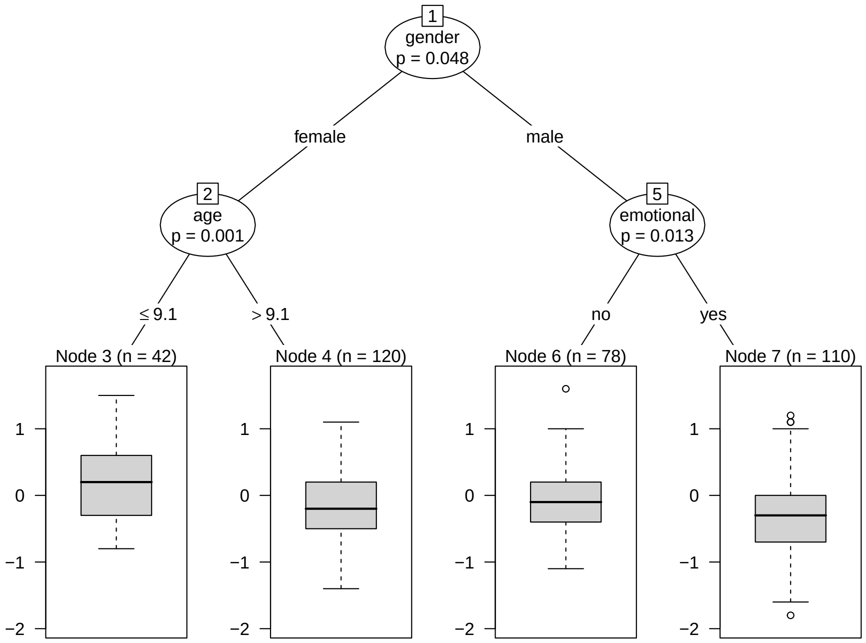

Figure 1: LMM tree for predicting treatment outcome.

By using argument `which`, we could have specified which part of the model should be plotted; by default, `which = "all"` the tree as well as the random effects are plotted.

The plotted tree is depicted in Figure 1. In every inner node of the plotted tree, the splitting variable and corresponding p-value from the parameter stability test is reported. To control for multiple testing, the p-values are Bonferroni corrected, by default. This can be turned off by adding `bonferroni = FALSE` to the function call, which yields a less conservative criterion for the parameter stability tests, but may increase the likelihood of overfitting. The significance level α equals .05 by default, but a different value may be specified by including `alpha = .01` in the function call, for example.

The Tree in Figure 1 shows the distribution of the observated values of the response in each of the terminal nodes. Four subgroups were found: terminal node 3 indicates that for female patients with lower age, the higher values for the response (i.e., poorer outcomes) were observed. Slightly better treatment outcomes are observed in terminal node 4 (females with higher age) and node 6 (males not presenting with emotional disorder). The best treatment outcomes are observed in node 7 (males presenting with emotional disorder).

Random-effects predictions are plotted in Figure 2. On average, patients at service provider 3 appear to have higher response variable values (poorer outcomes), while patients at service provider 10 appear to have more favorable outcomes.

To obtain numerical results, `print`, `coef`, codefixef, `ranef` and `VarCorr` methods are available (results omitted):

```
R> print(MH_tree)
R> coef(MH_tree)
```

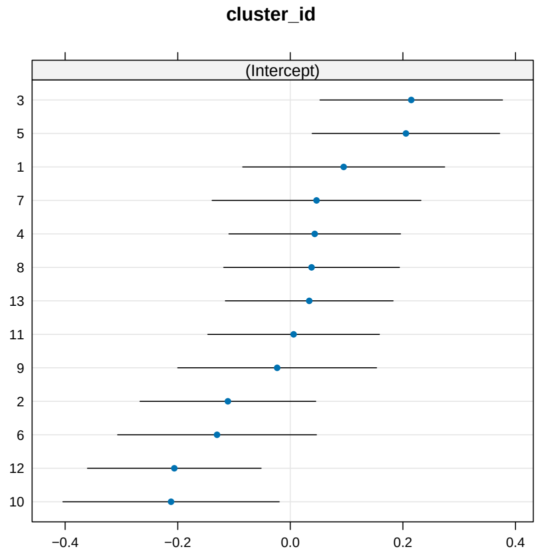

Figure 2: Predicted random effects predictions for the different service providers.

```
R> fixef(MH_tree)
R> ranef(MH_tree)
R> VarCorr(MH_tree)
```

To obtain predicted values, the `predict` method can be used:

```
R> predict(MH_tree, newdata = MHserviceDemo[1:10,])
```

```
          1           2           3           4           5 
 0.04161240 -0.27275670 -0.09567592 -0.13960661 -0.12668176 
          6           7           8           9          10 
-0.07843856 -0.28259716 -0.09567592 -0.09567592 -0.13546837 
```

If the `newdata` argument is not specified, predictions for the training observations are returned, by default. Also by default, the predictions are based on both random- and fixed-effects. Random effects can be excluded from the predictions by adding `re.form = NA`. This is useful, for example, when `newdata` specifies new observations whichare not part of a cluster from the training data (i.e., are from a 'new' cluster):

```
R> predict(MH_tree, newdata = MHserviceDemo[1:10, -7], re.form = NA)
```

```
          1           2           3           4           5 
-0.17325489 -0.31054321 -0.31054321 -0.17325489 -0.17325489 
          6           7           8           9          10 
-0.17325489 -0.07652208 -0.31054321 -0.31054321 -0.17325489 
```

2.1. Inspecting residuals

Residuals of the fitted mixed-effects tree can be obtained with the `residuals` method. Residuals can be used for assessing potential misspecification of the model or violation of assumptions (e.g., heteroscedasticity):

```
R> resids <- residuals(MH_tree)
R> preds <- predict(MH_tree)
R> plot(MHserviceDemo$cluster_id, resids)
R> scatter.smooth(preds, resids)
```

The plotted residuals are depicted in Figure 3. The left panel indicates some differences in residual variances across the levels of `cluster_id`, but these differences were not statistically significant (when tested with functions `bartlett.test()` or `fligner.test()`). The right panel of in Figure 3 indicates no pattern of association between fitted values and residuals.

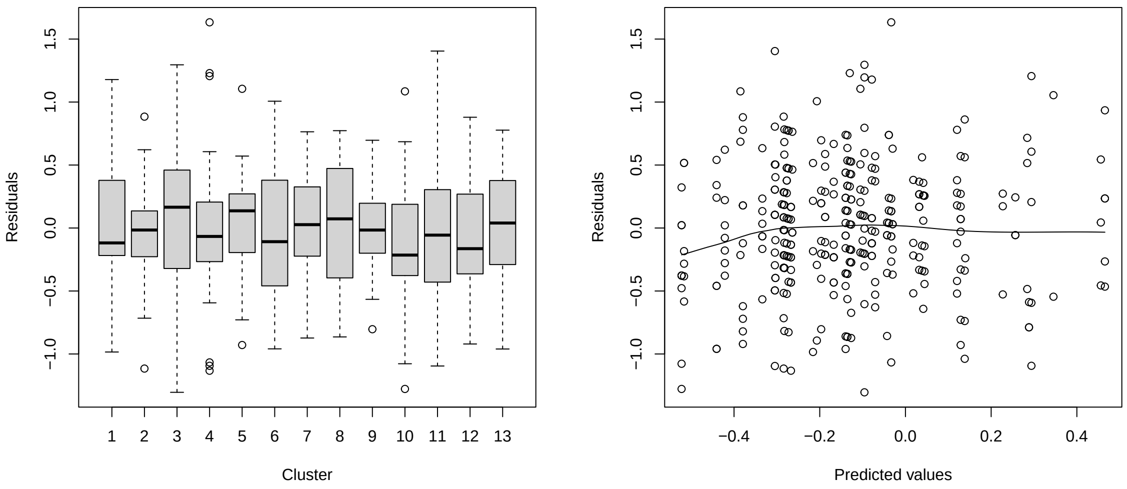

Figure 3: Residuals of the fitted linear mixed-effects model tree in Figure 1.

3. Detecting treatment-subgroup interactions in clustered data

In this example, we extend the model for the terminal nodes to accomodate a predictor variable: an indicator for treatment. Including predictor variables in the node-specific model may be particularly helpful when the interest is in detecting moderators. For example, in the detection of treatment-subgroup interactions where the effect of treatment may be moderated by one or more additional covariates. To illustrate, we will use the artificial motivating dataset from Fokkema *et al.* (2018), which can be recreated using the code provided in Appendix A, or can be loaded as follows:

```
R> data("DepressionDemo", package = "glmertree")
R> summary(DepressionDemo)
```

```
   depression          treatment      cluster        age      
 Min.   : 3.00   Treatment 1:78   1      :15   Min.   :18  
 1st Qu.: 7.00   Treatment 2:72   2      :15   1st Qu.:39  
 Median : 9.00                    3      :15   Median :45  
 Mean   : 9.12                    4      :15   Mean   :45  
 3rd Qu.:11.00                    5      :15   3rd Qu.:52  
 Max.   :16.00                    6      :15   Max.   :69  
                                  (Other):60               
    anxiety         duration      depression_bin
 Min.   : 3.00   Min.   : 1.000   0:78          
 1st Qu.: 8.00   1st Qu.: 5.000   1:72          
 Median :10.00   Median : 7.000                 
 Mean   :10.26   Mean   : 6.973                 
 3rd Qu.:12.00   3rd Qu.: 9.000                 
 Max.   :18.00   Max.   :17.000                 
```

The dataset includes seven variables: A continuous response variable (`depression`), a predictor variable for the linear model (`treatment`), three potential partitioning variables (`age`, `anxiety`, `duration`), an indicator for cluster (`cluster`) and a binarized response variable (`depression_bin`).

We fit a tree to the continuous response as follows:

```
R> lmmt <- lmertree(depression ~ treatment | cluster | age +
+                     duration + anxiety, data = DepressionDemo)
```

The left-hand side of the model formula (preceding the tilde symbol) specified the response variable (`depression`). The right-hand side of the model formula comprises three parts, separated by vertical bars: The first part specifies the predictor variable(s) of the node specific (G) LMM (`treatment`, in this example). The second part specifies the predictors of the global model (comprising only a random intercept with respect to `cluster`, in this example). The third part specifies the potential partitioning variables. In this example, all partitioning variables are continuous, but (ordered) categorical partitioning variables can also be specified.

We specified only a single variable for the global model, resulting in estimation of a random intercept with respect to `cluster`. By default, if only a single variable is specified for the global model, a random intercept with respect to the specified variable will be estimated. More complex random effects can be specified using the customary formulation employed in **lme4**. For example, we could specify a global model comprising a correlated random intercept and slope of `age`, both estimated with respect to `cluster`:

```
R> depression ~ treatment | (age + (1 + age | cluster)) | age +
+    duration + anxiety
```

Note that we included a fixed effect for `age` in the global model, as is customary when specifying a random effect. Note also that we used round brackets in order to protect the vertical bars in the formulation of the (global) random effects.

We could also encounter nested multilevel structures, for example when we have patients from treatment centers nested within geographical areas. Using the indicators for treatment center and geographical areas (e.g., `center` and `area`), we could have specified random intercept terms for `center`, nested within `area`:

```
R> depression ~ treatment | (1|center/area) | age + duration + anxiety
```

```
depression ~ treatment | (1 | center/area) | age + duration +
    anxiety
```

Using the `plot` method, we can plot the resulting tree and random effects:

```
R> plot(lmmt)
```

The plotted tree is depicted in Figure 4. In every inner node of the plotted tree, the splitting variable and corresponding *p*-value from the parameter stability test is reported. To control for multiple testing, the *p*-values are Bonferroni corrected, by default.

The plotted tree in Figure 4 shows that there are three subgroups with differential treatment effectiveness: node 3 indicates that for patients with lower duration and lower anxiety, Treatment 1 leads to lower post-treatment depression. Node 4 indicates that for patients with lower duration and higher anxiety, both treatments yield more or less the same expected outcome. Node 5 indicates, that for patients with higher duration, Treatment 2 leads to lower post-treatment depression.

The predicted random effects are plotted in Figure 5. On average, patients from cluster 10 have somewhat higher expected post-treatment depression scores, whereas patients from cluster 4 have somewhat lower expected post-treatment depression scores.

Alternatively, we can request caterpillar plots of the estimated node-specific coefficients through specifying `which = "tree.coef"`:

```
R> plot(lmmt, which = "tree.coef")
```

The plotted results depict the node-specific parameter estimated with error bars (±1.96 times the standard error). Note that these standard errors do not account for the searching of the tree structure and are likely too small, but they do allow for gauging the precision of the estimated parameters.

To obtain numerical results, `print`, `coef`, `fixef`, `ranef`, and `VarCorr` methods are available (results omitted):

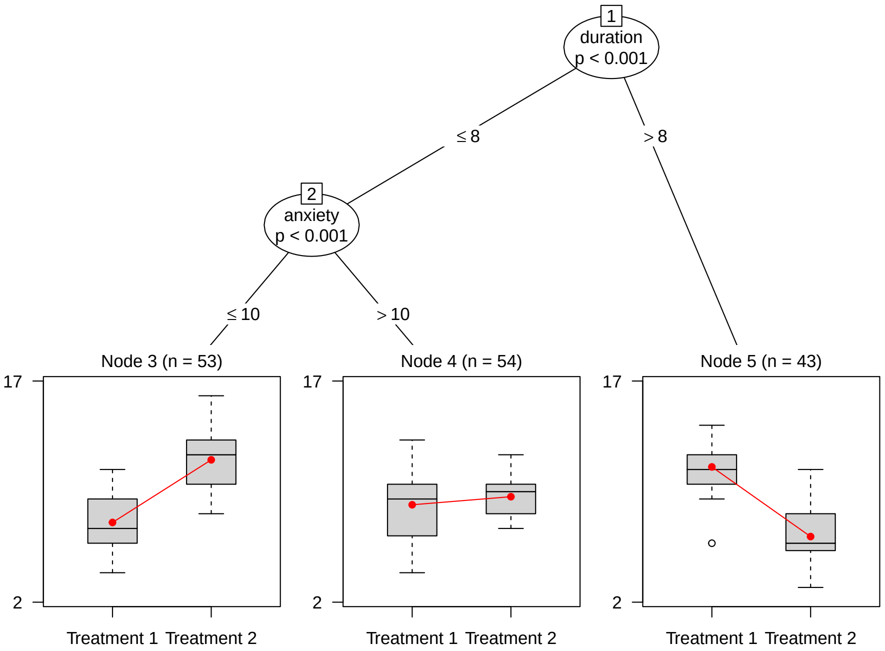

Figure 4: Linear mixed-effects model tree with treatment-subgroup interactions.

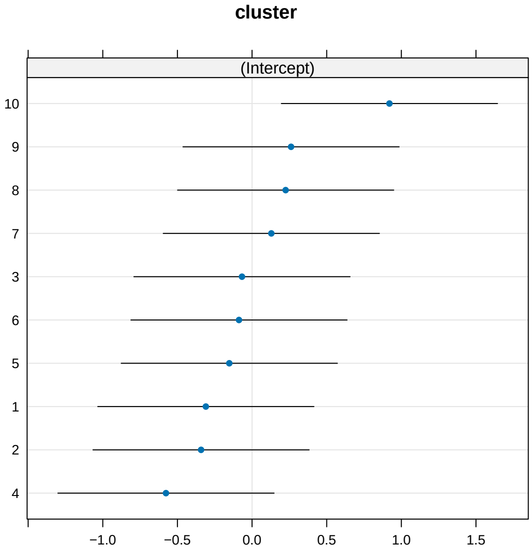

Figure 5: Random effects for the linear mixed-effects model tree in Figure 4.

```
R> print(lmmt)
R> coef(lmmt)
R> fixef(lmmt)
R> ranef(lmmt)
```

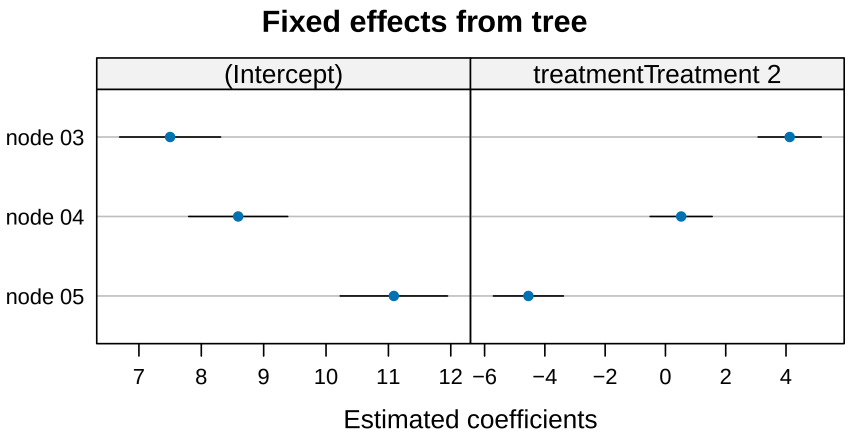

Figure 6: Estimated node-specific fixed effects with error bars.

```
R> VarCorr(lmmt)
```

To obtain predicted values, the `predict` method can be used:

```
R> predict(lmmt, newdata = DepressionDemo[1:7,])

        1         2         3         4         5         6         7 
10.777968 11.554672  7.158595  9.045117 11.280677  8.816418 11.883480 
```

When `newdata` is not specified, predictions for the training observations are returned, by default. Random effects can be excluded from the predictions by adding `re.form = NA`. This is useful, for example, when `newdata` is specified, but the new observations do not have a cluster indicator or are from new clusters:

```
R> predict(lmmt, newdata = DepressionDemo[1:7, -3], re.form = NA)

        1         2         3         4         5         6         7 
11.087612 11.622223  7.500140  9.112668 11.622223  8.591409 11.622223 
```

3.1. Inspecting residuals

Residuals of the fitted (G)LMM tree can be obtained with the `residuals` method. This can be useful for assessing potential misspecification of the model (e.g., heteroscedasticity):

```
R> resids <- residuals(lmmt)
R> preds <- predict(lmmt)
R> plot(factor(DepressionDemo$cluster), resids)
R> scatter.smooth(preds, resids)
```

The plotted residuals are depicted in Figure 7. The left panel indicates some variation in error variances across levels of the random effects, but it appears these differences are not statistically significant:

```
R> fligner.test(resids ~ DepressionDemo$cluster)

        Fligner-Killeen test of homogeneity of variances

data:  resids by DepressionDemo$cluster
Fligner-Killeen:med chi-squared = 9.0313, df = 9, p-value =
0.4344

R> bartlett.test(resids ~ DepressionDemo$cluster)

        Bartlett test of homogeneity of variances

data:  resids by DepressionDemo$cluster
Bartlett's K-squared = 5.6663, df = 9, p-value = 0.7728
```

The right panel of Figure 7 shows fitted values against residuals and also does not reveal a pattern indicating model misspecification.

4. Detecting subgroups with different growth trajectories

An artificially generated, longitudinal dataset is included in package **glmertree** and can be loaded as follows:

```
R> data("GrowthCurveDemo", package = "glmertree")
R> dim(GrowthCurveDemo)

[1] 1250   11
```

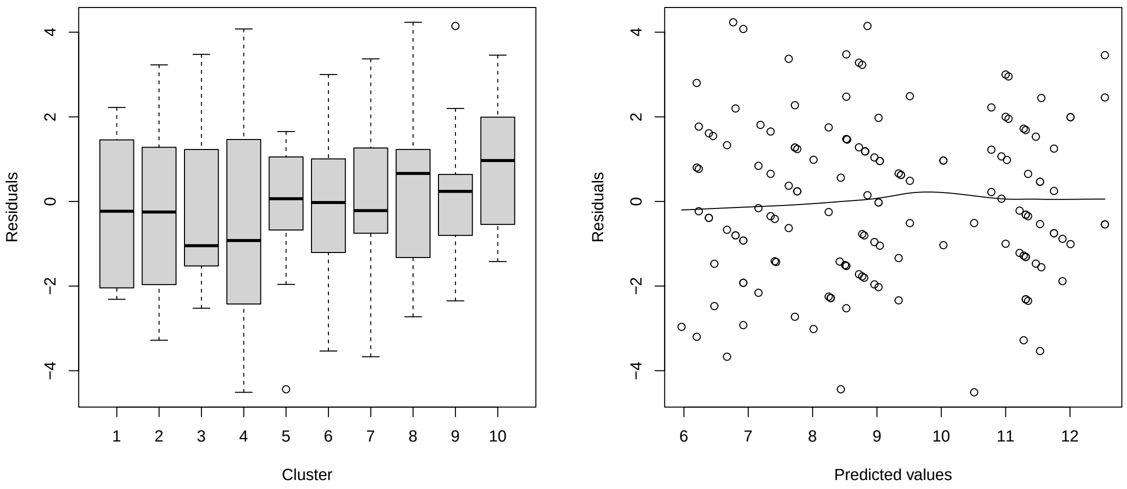

Figure 7: Residuals of the fitted linear mixed-effects model tree in Figure 4.

```
R> names(GrowthCurveDemo)
```

```
 [1] "x1"     "x2"     "x3"     "x4"     "x5"     "x6"     "x7"
 [8] "x8"     "person" "time"   "y"
```

The dataset contains 1250 repeated measurements from 250 individuals. The data is in long format and the response was measured at five timepoints for each individual. The dataset contains 11 variables: A continuous response variable (`y`), a predictor variable for the linear model (`time`, taking values 0 through 4), time-invariant potential partitioning variables (`x1` through `x8`), and an indicator for person (`person`).

The data were generated so that `x1`, `x2` and `x3` are true partitioning variables. Furthermore, `x1` is a binary variable, while all other potential partitioning variables follow a normal distribution with $\mu = 0$ and $\sigma = 5$. Potential partitioning variables were generated so as to be uncorrelated. Random intercepts and slopes were generated so that the intercept and slope values for persons vary around their node-specific means, following a normal distribution with $\mu = 0$ and $\sigma = \sqrt{2}$ for the intercept and $\sigma = \sqrt{.4}$ for the slope. Errors were uncorrelated and followed a normal distribution with $\mu = 0$ and $\sigma = \sqrt{5}$.

The default fitting procedure as employed by functions `lmertree()` and `glmertree()` assume potential predictor variables are measured on the observation level. In this example, potential partitioning variables are measured on the cluster level (i.e., time-invariant covariates) and the observation-level stability tests will likely have inflated type-I error. We can account for the level of the partitioning variables through specification of the `cluster` argument. As a result, parameter stability tests will be performed on the cluster instead of the observation level:

```
R> gc_tree <- lmertree(y ~ time | person | x1 + x2 + x3 + x4 + x5 + x6 +
+                          x7 + x8, cluster = person, data = GrowthCurveDemo)
```

The first part of the formula (`y ~ time`) regresses the response on time. The second part (`| person |`) specifies that a random intercept should be estimated with respect to `person`. The third part (`x1 + ... + x8`) specifies the potential partitioning variables. Using the cluster-level stability tests, we obtained a tree with four subgroups (terminal nodes):

```
R> width(gc_tree$tree)
```

```
[1] 4
```

Employing the default observation-level stability tests would have yielded a tree with more (possibly spurious) subgroups:

```
R> gc_obs_tree <- lmertree(y ~ time | person | x1 + x2 + x3 + x4 + x5 +
+                              x6 + x7 + x8, data = GrowthCurveDemo)
R> width(gc_obs_tree$tree)
```

```
[1] 10
```

We can plot the growth-curve tree using the `plot` method:

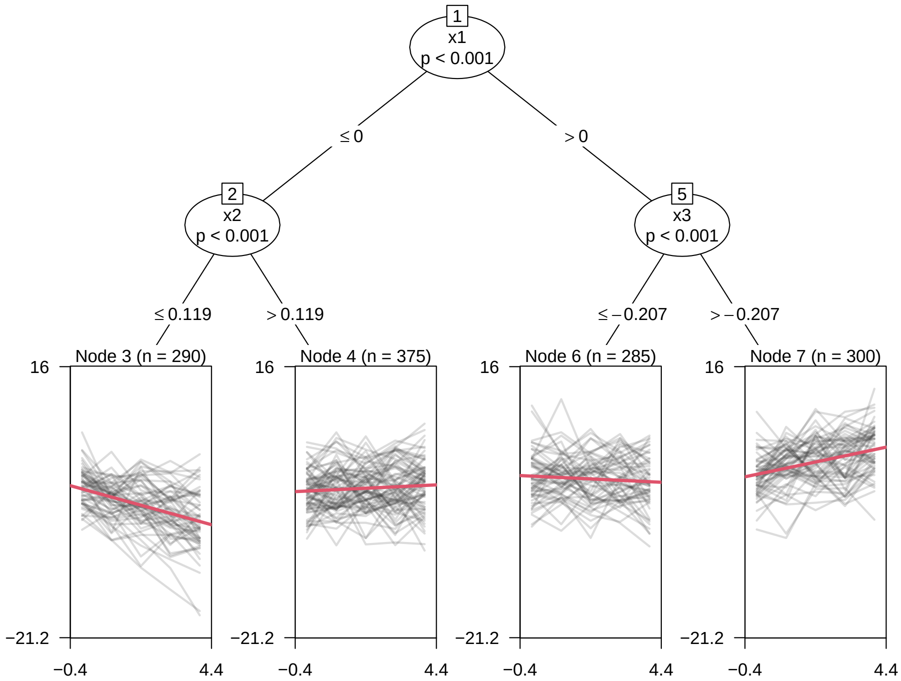

Figure 8: Linear mixed-effects model tree with growth curve models in the terminal nodes.

```
R> plot(gc_tree, which = "tree", fitted = "marginal")
```

Note that we additionally specified `which = "tree"`, to obtain a plot of the tree only, and `fitted = "marginal"`. The latter specified that the fitted values (represented by the red lines in the terminal nodes) should be computed by fixing all remaining (fixed- an random-effects) predictor variables at their means (or majority class, for categorical predictors). By default, `fitted = "combined"` yields fitted values, computed based on the observed values of the remaining (random and fixed-effects) predictor variables. We employed the marginal approach here, as this yields straight lines plotted in the terminal nodes, which may better reflect the average trajectories. To depict observed data as growth curves instead of single datapoints, specify ‘which = "growth"‘:

```
R> plot(gc_tree, which = "growth")
```

The resulting plot is depicted in Figure 8. The red lines in the terminal nodes represent the average trajectory within the terminal nodes. The dots represent the observed data values.

The plot reveals that the true partitioning variables (`x1`, `x2` and `x3`) were selected for splitting. The fitted models in the terminal nodes (red lines) reveal a decrease in the response variable over time for the left-most subgroup, and an increase for the right-most subgroup. The curves in the two middle subgroups are rather flat, indicating no change over time. We can also print the values of the estimated coefficients in the terminal nodes:

```
R> plot(gc_tree, type = "simple", which = "tree")
```

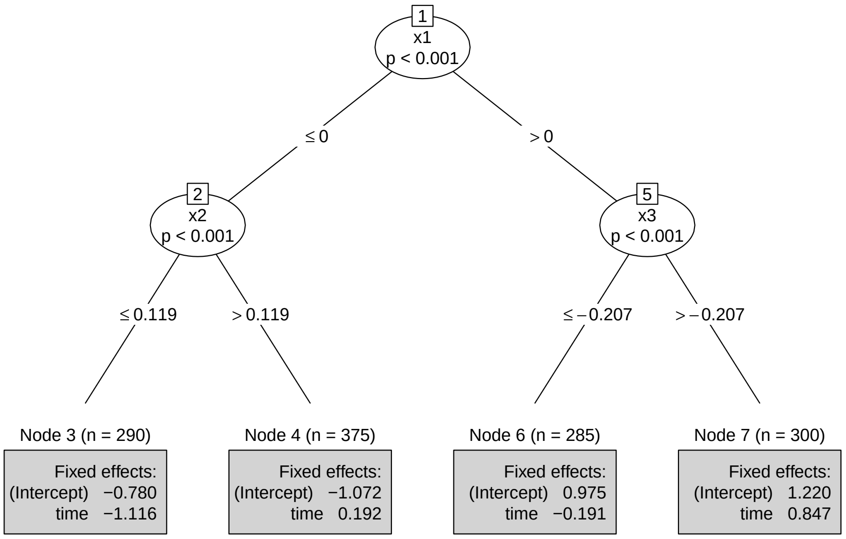

Figure 9: Linear mixed-effects model tree with estimated coefficients printed in the terminal nodes.

The tree in Figure 9 reveals effects of time relatively close to zero for nodes 4 and 6. We can also plot the coefficients with error bars:

```
R> plot(gc_tree, which = "tree.coef")
```

The standard errors used for creating the error bars in Figure 10 do not account for the searching of the tree sturcture and may be too small. However, the overlapping error bars for the effect of time in nodes 6 and 4 indicate a non-significant difference between the effects of time in these nodes. The observed data points in Figure 8 indicate that the individual observations show substantial variation around the estimated fixed effects. To obtain an estimate of the random effects and residual variances, we can use the `VarCorr` method:

```
R> varcor <- VarCorr(gc_tree)
R> varcor

 Groups   Name        Std.Dev.
 person   (Intercept) 2.2449
 Residual             2.3696
```

WE can use these variances to obtain an estimate of the intraclass correlation (ICC):

```
R> res_var <- attr(varcor, "sc")^2
R> int_var <- as.numeric(varcor$person)
```

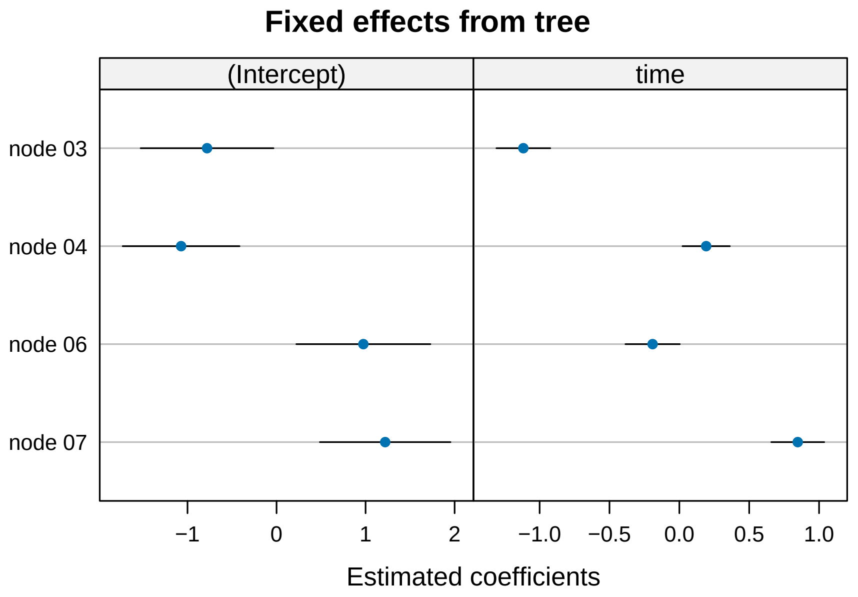

Figure 10: Caterpillar plots of estimated coefficients for each of the terminal nodes.

```
R> ICC <- int_var / (res_var + int_var)
R> ICC

[1] 0.4729834
```

The value of the ICC indicates that about 47.3 percent of variance in the response is accounted for by inter-individual variation.

4.1. Adding a random slope of time

Earlier, we specified a model formula with only a random intercept and thus did not account for possible variation between persons in the effect of time, within terminal nodes. To account for such differences we can incorporate a random slope of time into the model formula:

```
R> form_s <- formula(paste0("y ~ time | (1 + time | person) | ",
+                          paste0("x", 1:8, collapse = " + ")))
R> form_s

y ~ time | (1 + time | person) | x1 + x2 + x3 + x4 + x5 + x6 +
    x7 + x8
```

Again, we fit the tree:

```
R> gc_tree_s <- lmertree(form_s, cluster = person, data = GrowthCurveDemo)
```

In this case, we obtained the same tree structure with or without estimating random slopes (Figure 8). This need not necessarily be the case with other datasets. At the very least, the estimated random effects can provide us with additional information about variation due to between-person differences in initial levels and growth over time:

```
R> VarCorr(gc_tree_s)
```

```
 Groups   Name        Std.Dev. Corr  
 person   (Intercept) 2.06706        
          time        0.58917  -0.092
 Residual             2.18126        
```

Compared to the fitted model with random intercepts only, we see that the residual variance decreased somewhat.

References

Bates D, Mächler M, Bolker B, Walker S (2015). "Fitting Linear Mixed-Effects Models Using **lme4**." doi:10.18637/jss.v067.i01.

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). doi:10.3758/s13428-017-0971-x.

Hothorn T, Zeileis A (2015). "**partykit**: A Modular Toolkit for Recursive Partytioning in R." *Journal of Machine Learning Research*, **16**, 3905–3909. URL http://www.jmlr.org/papers/v16/hothorn15a.html.

A. R code for generating artificial motivating dataset

Generate the predictor variables and error term:

```
R> set.seed(123)
R> treatment <- rbinom(n = 150, size = 1, prob = .5)
R> duration <- round(rnorm(150, mean = 7, sd = 3))
R> anxiety <- round(rnorm(150, mean = 10, sd = 3))
R> age <- round(rnorm(150, mean = 45, sd = 10))
R> error <- rnorm(150, 0, 2)
```

Generate the random intercepts:

```
R> cluster <- error + rnorm(150, 0, 6)
R> rand_int <- sort(rep(rnorm(10, 0, 1), each = 15))
R> rand_int[order(cluster)] <- rand_int
R> error <- error - rand_int
R> cluster[order(cluster)] <- rep(1:10, each = 15)
```

Generate treatment subgroups:

```
R> node3t1 <- ifelse(duration <= 8 & anxiety <= 10 & treatment == 0, -2, 0)
R> node3t2 <- ifelse(duration <= 8 & anxiety <= 10 & treatment == 1, 2, 0)
R> node5t1 <- ifelse(duration > 8 & treatment == 0, 2.5, 0)
R> node5t2 <- ifelse(duration > 8 & treatment == 1, -2.5, 0)
```

Generate the continuous and dichotomized outcome variable:

```
R> depression <- round(9 + node3t1 + node3t2 + node5t1 + node5t2 +
+     .4 * treatment + error + rand_int)
R> depression_bin <- factor(as.numeric(depression > 9))
```

Make treatment indicator a factor and collect everything in a data frame:

```
R> treatment <- factor(treatment, labels = c("Treatment 1", "Treatment 2"))
R> DepressionDemo <- data.frame(depression, treatment, cluster,
+    age, anxiety, duration, depression_bin)
```

Affiliation:

Marjolein Fokkema
Department of Methods & Statistics, Intitute of Psychology
Universiteit Leiden
Wassenaarseweg 52
2333 AK Leiden, The Netherlands
E-mail: m.fokkema@fsw.leidenuniv.nl
URL: http://www.marjoleinfokkema.nl/

Achim Zeileis
Department of Statistics
Faculty of Economics and Statistics
Universität Innsbruck
Universitätsstr. 15
6020 Innsbruck, Austria
E-mail: Achim.Zeileis@R-project.org
URL: https://eeecon.uibk.ac.at/~zeileis/