Fitting Generalized Linear Mixed-Effects Model Trees

Marjolein Fokkema
Universiteit Leiden

Achim Zeileis
Universität Innsbruck

Abstract

This vignette briefly introduces the **glmertree** package for fitting a wide range of generalized linear mixed-effects model trees (GLMM trees or glmertrees). In hands-on examples baed on artificial datasets, emphasis is given to the special cases of fitting regression trees with constant fits in the terminal nodes to clustered data (Section 2), detecting treatment-subgroup interactions in clustered data (Section 3), and detecting subgroups in linear 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 recently been proposed by Fokkema, Smits, Zeileis, Hothorn, and Kelderman (2018) for detecting treatment-subgroup interactions in clustered datasets. Using a hands-on (artificial) example, this vignette describes how to fit such GLMM trees: Section 3 will describe how to assess main and interaction effects of a categorical variable (treatment) on a continuous response (treatment outcome). But first, Section 2 will describe how to fit (G)LMM trees with constant fits in the terminal nodes. The R package **glmertree** can be used to detect predictors and moderators in a wide range of generalized linear mixed-effects models.

GLMM trees estimate a global random-effects model, using all training observations. The fixed-effects model is estimated locally: the dataset is partitoned with respect to additional covariates or partitioning variables and a fixed-effects model is estimated in each cell of the partition. The **glmertree** package makes use of the **partykit** package (Hothorn and Zeileis 2015) to find the partition and the **lme4** package (Bates, Mächler, Bolker, and Walker 2015) to fit the mixed-effects model.

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

```
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 the **glmertree** package are `lmertree()`, for continuous outcome variables, and `glmertree()`, for binary or count outcome variables.

2. Fitting regression trees to clustered data

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, where higher values reflect 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 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 main functions in the **glmertree** package are `lmertree()`, for continuous outcome variables, and `glmertree()`, for binary or count responses. Both functions require the user to specify at least two arguments: `formula` and `data`.

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

The first argument specified the model formula. The left hand side (preceding the tilde), specifies the response variable, which is `outcome` in the current example. The right-hand side of the model formula (following the tile) consists of three parts, separated by vertical bars: The

first part specifies the subgroup-specific fixed-effect model, consisting only of an intercept (`1`) in the current example. The second part specifies the random effects, consisting of only a single variable, resulting in estimation of a random intercept with respect to `cluster_id`. Finally, the third part specifies the potential partitioning variables: `age`, `gender`, `emotional`, `autism`, `impact` and `conduct`. A more complex random-effects structures could also be specified. For example, specifying the model formula as:

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

would yield a model in which a random intercept and slope for `age` would be estimated with respect to `cluster_id`. Note that the parentheses are necessary to protect the vertical bars, which separate the random effects from the other parts of the model.

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:

```
R> MHserviceDemo$outcome_bin <- factor(outcome > 0)
R> glmmt <- 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(lmmt)
```

Using the argument `which`, we can also specify which part of the model should be plotted: `which = "tree"` plots only the tree, `which = "ranef"` plots only the predicted random effects and `which = "all"` (the default) plots the tree as well as the random effects.

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, yielding a less conservative criterion for the parameter stability tests, but note that this will increase the likelihood of overfitting. The significance level α equals .05 by default, but a different value, say for example .01, can be specified by including `alpha = .01` in the function call.

The plotted tree shows that there are four subgroups: node 3 indicates that for female patients with lower age, somewhat higher values for the response are observed and thus for these patients, poorer treatment outcomes are predicted. Somewhat better treatment outcomes are observed for those in node 4 (with female gender and higher age) and node 6 (male gender and no emotional disorder). The best treatment outcomes (lowest response variable values) are observed among those in node 7 (male gender and presence of an emotional disorder).

The predicted random effects 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 for space considerations):

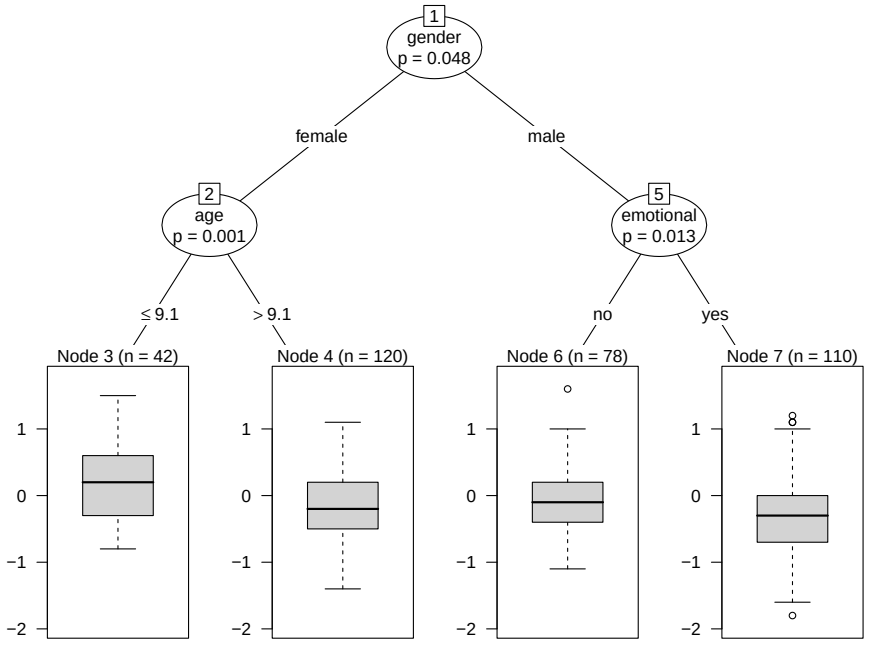

Figure 1: Linear mixed-effects model tree with constant fits in the terminal nodes.

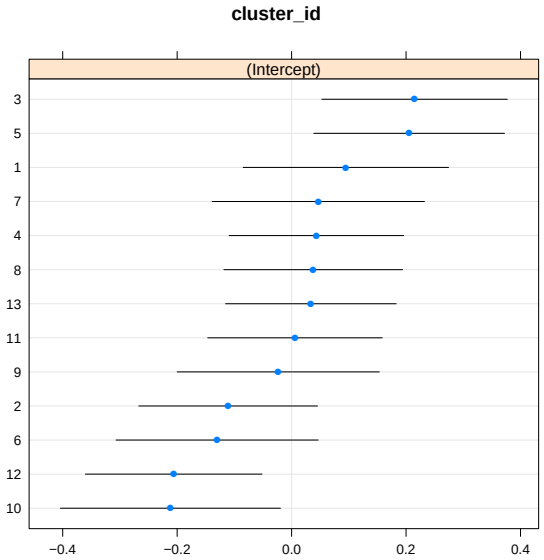

Figure 2: Random effects for the linear mixed-effects model tree in Figure 1.

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

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

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

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

```
          1           2           3           4           5
 0.46541482 -0.03873556  0.13834522  0.28419580  0.29712065
          6           7           8           9          10
 0.34536386 -0.37932997  0.13834522  0.13834522  0.28833404
```

When `newdata` 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 (tree) predictions. 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 = MHserviceDemo[1:10, -7], re.form = NA)
```

```
          1           2           3           4           5
 0.25054752 -0.07652208 -0.07652208  0.25054752  0.25054752
          6           7           8           9          10
 0.25054752 -0.17325489 -0.07652208 -0.07652208  0.25054752
```

2.1. Inspecting residuals

Residuals of the fitted mixed-effects 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(MHserviceDemo$cluster_id, resids)
R> scatter.smooth(preds, resids)
```

The plotted residuals are depicted in Figure 3. The left panel indicates there may be some differences in the error variances across the levels of `cluster_id`. The right panel indicates no association between fitted values and residuals.

3. Detecting treatment-subgroup interactions in clustered data

The (generalized) linear model specified for the terminal nodes can easily be extended to accomodate additional predictor variables. This may be particularly helpful when the interest is in the detection of moderators. For example, in the detection of treatment-subgroup interactions, where the effect of treatment on the response variable may be moderated by one or more additional covariates. To illustrate, we will use an 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   Min.   : 1.0   Min.   :18
 1st Qu.: 7.00   Treatment 2:72   1st Qu.: 3.0   1st Qu.:39
 Median : 9.00                    Median : 5.5   Median :45
 Mean   : 9.12                    Mean   : 5.5   Mean   :45
 3rd Qu.:11.00                    3rd Qu.: 8.0   3rd Qu.:52
 Max.   :16.00                    Max.   :10.0   Max.   :69
    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 the model as follows:

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

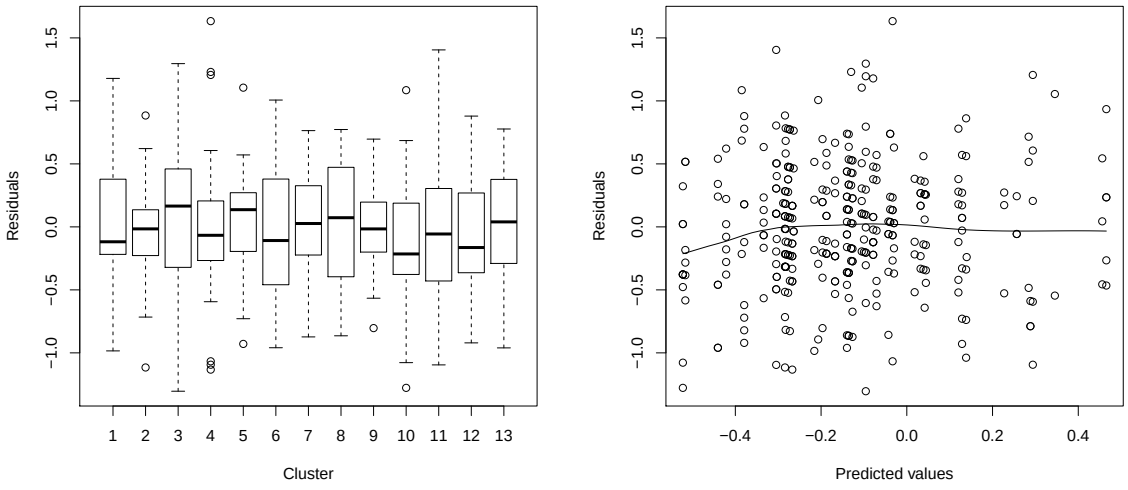

Figure 3: Residuals of the fitted linear mixed-effects model tree in Figure 1.

The left hand side of the model formula (preceding the tilde symbol) specifies the response variable (`depression`). The right hand side of the model formula consists of three parts, separated by vertical bars: The first part specifies the predictor variable(s) of the (generalized) linear model (`treatment`, in this example). The second part specifies the random effects and the third part specifies the potential partitioning variables. All partitioning variables are continuous in this example, but note that (ordered) categorical partitioning variables may also be specified. Also, we specified a single variable in the random-effects part, resulting in estimation of a random intercept with respect to `cluster`. More complex random effects can also be specified; for example, specifying the model formula as

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

would yield a model with a random intercept and slope for `age` estimated with respect to `cluster`. The brackets are necessary to protect the vertical bars in the formulation of the random effects.

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 the dichotomized response `depression_bin` could be obained by:

```
R> glmm_tree <- glmertree(depression_bin ~ treatment | cluster |
+    age + duration + anxiety, data = DepressionDemo, family = binomial)
```

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

```
R> plot(lmm_tree)
```

Using the argument `which`, we can also specify which part of the model should be plotted: `which = "tree"` plots only the tree, `which = "ranef"` plots only the predicted random effects and `which = "all"` (the default) plots the tree as well as the random effects.

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. This can be turned off by adding `bonferroni = FALSE` to the function call, yielding a less conservative criterion for the parameter stability tests, but note that this will increase the likelihood of overfitting. The significance level α equals .05 by default, but a different value, say for example .01, can be specified by including `alpha = .01` in the function call.

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.

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

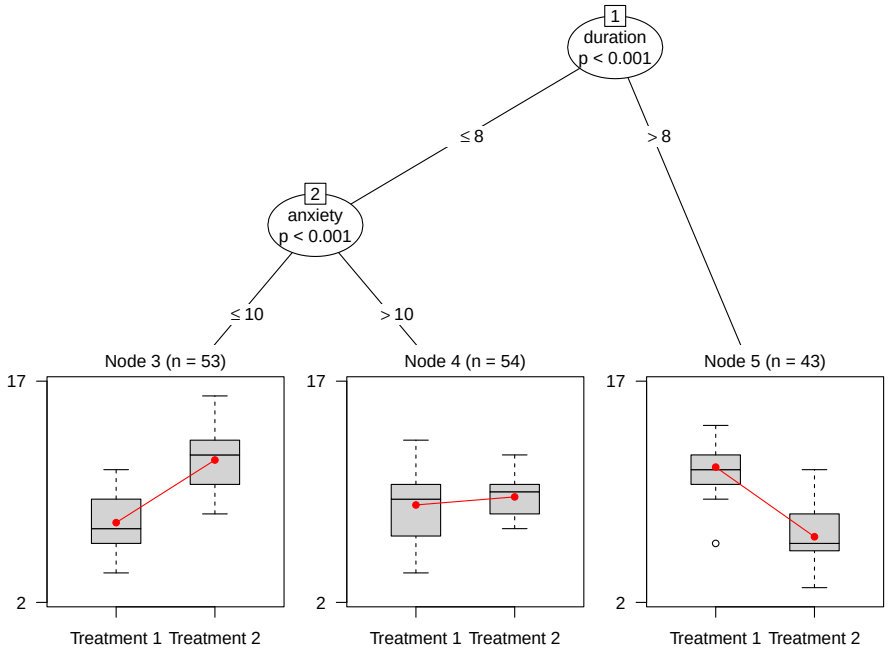

Figure 4: Linear mixed-effects model tree with treatment-subgroup interactions.

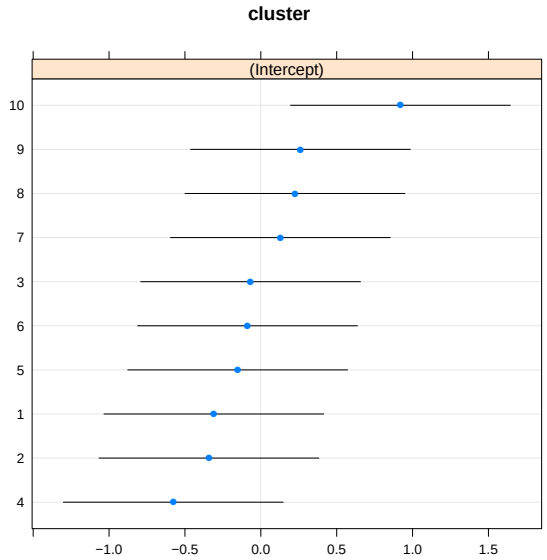

Figure 5: Random effects for the linear mixed-effects model tree in Figure 4.

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

```
R> VarCorr(lmm_tree)
```

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

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

       1        2        3        4        5        6        7
10.777967 11.554671  7.158594  9.045116 11.280676  8.816419 11.883481
```

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(lmm_tree, newdata = DepressionDemo[1:7, -3], re.form = NA)

       1        2        3        4        5        6        7
11.087612 11.622223  7.500141  9.112668 11.622223  8.591409 11.622223
```

3.1. Inspecting residuals

Residuals of the fitted GLMM 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(lmm_tree)
R> preds <- predict(lmm_tree)
R> plot(factor(DepressionDemo$cluster), resids)
R> scatter.smooth(preds, resids)
```

The plotted residuals are depicted in Figure 6. The left panel does not indicate substantial variation in error variances across levels of the random effects. The right panel show fitted values plotted against residuals and also does not reveal a pattern indicating model misspecification.

4. Detecting subgroups in linear growth curve models

For partitioning longitudinal data, function `lmertree()` requires data to be in the long format. An artificially generated longitudinal dataset is included in the package and can be loaded as follows:

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

[1] 1250   31

R> names(GrowthCurveDemo)
```

```
 [1] "x1"     "x2"     "x3"     "x4"     "x5"     "x6"     "x7"
 [8] "x8"     "x9"     "x10"    "x11"    "x12"    "x13"    "x14"
[15] "x15"    "x16"    "x17"    "x18"    "x19"    "x20"    "x21"
[22] "x22"    "x23"    "x24"    "x25"    "x26"    "x27"    "x28"
[29] "person" "time"   "y"
```

The dataset contains 1250 repeated measurements from 250 individuals. The response was measured at five timepoints for each individual. The dataset contains 31 variables: A continuous response variable (`y`), a predictor variable for the linear model (`time`, taking values 0 through 4), 28 potential partitioning variables (`x1` through `x28`), 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 $\sigma^2 = 25$. 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 $\sigma^2 = 2$ for the intercept and $\sigma^2 = .4$ for the slope. Errors were uncorrelated and followed a normal distribution with $\mu = 0$ and $\sigma^2 = 5$.

Because we have a relatively large amount of potential partitioning variables, we first construct the model formula as follows:

```
R> form <- formula(paste0("y ~ time | person | ",
+                         paste0("x", 1:28, collapse = " + ")))
R> form
```

```
y ~ time | person | x1 + x2 + x3 + x4 + x5 + x6 + x7 + x8 + x9 +
    x10 + x11 + x12 + x13 + x14 + x15 + x16 + x17 + x18 + x19 +
    x20 + x21 + x22 + x23 + x24 + x25 + x26 + x27 + x28
```

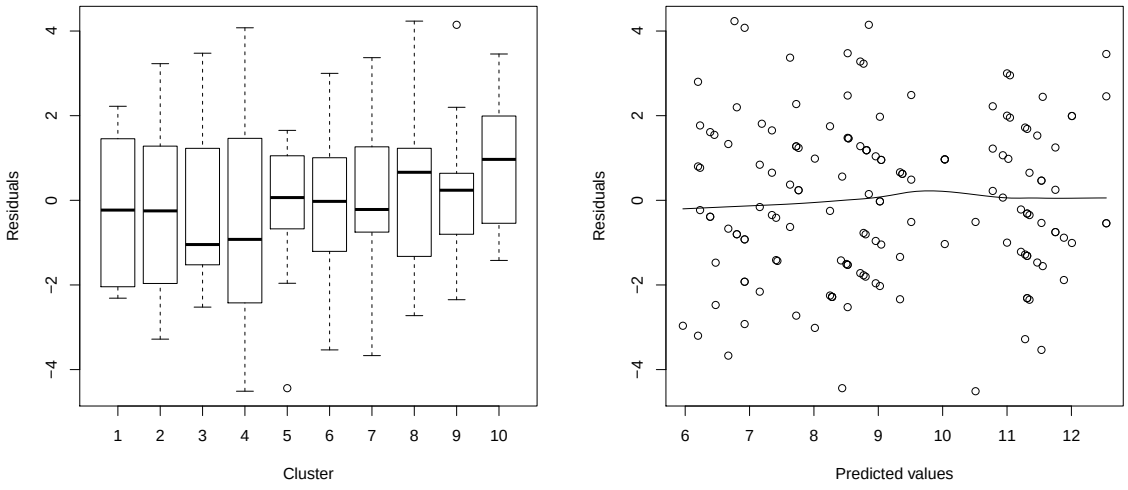

Figure 6: Residuals of the fitted linear mixed-effects model tree in Figure 4.

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 + ... + x28`) specifies the potential partitioning variables.

The default fitting procedure as employed in `lmertree()` will assumes potential predictor variables are measured on the observation level. With longitudinal data in the long format, and with potential partitioning variables measured on the cluster level (i.e., time-invariant covariates), the observation-level stability tests will likely have inflated type-I error. This can be accounted for through specification of the `cluster` argument. As a result, parameter stability tests will be performed on the cluster instead of the observation level:

```
R> gcm_tree <- lmertree(form, cluster = person, data = GrowthCurveDemo)
```

Using the cluster-level stability tests yields a tree with 4 subgroups (terminal nodes):

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

```
[1] 4
```

Employing the default observation-level stability tests would have yielded a tree with many spurious splits and subgroups:

```
R> gcm_obs_tree <- lmertree(form, data = GrowthCurveDemo)
R> width(gcm_obs_tree$tree)
```

```
[1] 17
```

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

```
R> plot(gcm_tree, which = "tree")
```

By default, the fixed effect of the predictor variable in the linear model (in this case, `time`) is plotted in 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.

The observed data points 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(gcm_tree)
R> varcor
```

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

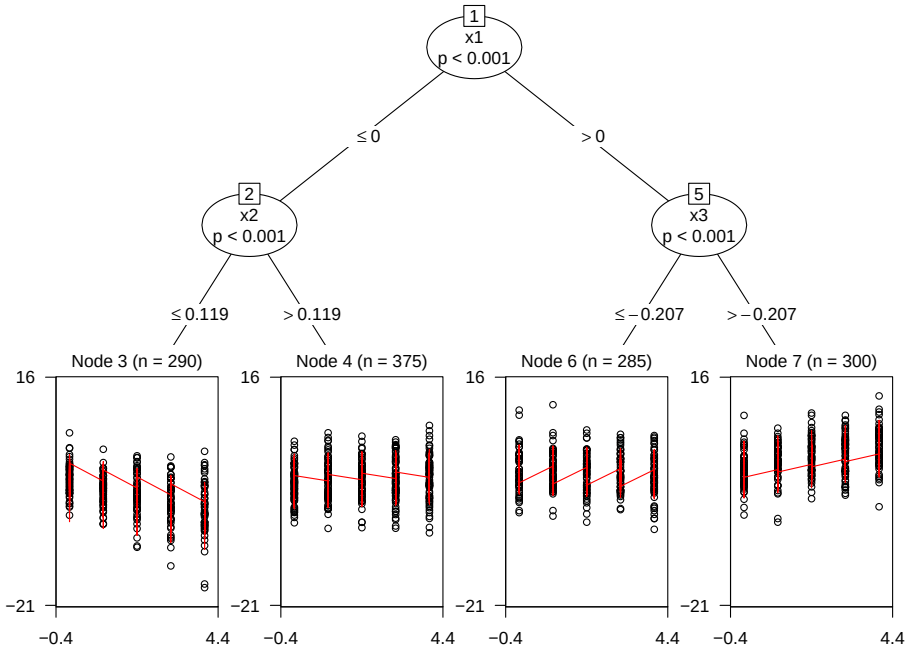

Figure 7: Linear mixed-effects model tree with growth curve models in the terminal nodes.

To obtain an estimate of the intraclass correlation (ICC), we could divide the variance of the random intercept by the variance of the residuals and that of the random intercept:

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

```
[1] 0.4729834
```

4.1. Random slopes

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", 1:28, collapse = " + ")))
R> form_s
```

```
y ~ time | (1 + time | person) | x1 + x2 + x3 + x4 + x5 + x6 +
    x7 + x8 + x9 + x10 + x11 + x12 + x13 + x14 + x15 + x16 +
    x17 + x18 + x19 + x20 + x21 + x22 + x23 + x24 + x25 + x26 +
    x27 + x28
```

Again, we fit the tree:

```
R> gcm_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 7). 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(gcm_tree_s)
```

```
 Groups   Name        Std.Dev. Corr
 person   (Intercept) 2.0671
          time        0.5892   -0.092
 Residual             2.1812
```

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). URL https://link.springer.com/article/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/