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 (artificial) examples, emphasis is given to the special cases of fitting GLMM trees with constant fits in the terminal nodes and detecting treatment-subgroup interactions in clustered data.

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. In what follows, we will focus on the special cases of fitting mixed-effects trees with constant fits in the terminal nodes to clustered data and detection of treatment-subgroup interactions in clustered data.

2. Fitting (G)LMM trees with constant fits in the terminal nodes

For this example, we will make use of an artificially generated dataset of $N = 3,739$ young people who received treatment at one of 13 mental-health service providers. Potential predictor variables are demographic variables and case characteristics. The response variable is a treatment outcome, as measured by a mental-health difficulties score at follow-up, corrected for the baseline assessment, where higher values relect poorer treatment outcome. The dataset includes two continuous covariates (`age` and `impact`); four binary covariates (`gender`, `emotional`, `autism` and `conduct`), an indicator for service provider (`cluster_id`) and a continuous variable reflecting treatment outcome (`outcome`):

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

```
      age             impact          gender     emotional  autism
 Min.   : 4.80   Min.   :-8.300   female:1837   no :1692   no :3409
 1st Qu.: 9.20   1st Qu.: 2.200   male  :1902   yes:2047   yes: 330
 Median :11.30   Median : 4.200
 Mean   :11.46   Mean   : 4.149
 3rd Qu.:13.60   3rd Qu.: 6.000
 Max.   :23.60   Max.   :13.500

 conduct      cluster_id      outcome
 no :2979   13     : 308   Min.   :-1.9000
 yes: 760   6      : 300   1st Qu.:-0.5000
            2      : 299   Median :-0.1000
            12     : 296   Mean   :-0.1271
            7      : 295   3rd Qu.: 0.2000
            3      : 293   Max.   : 1.8000
            (Other):1948
```

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

With the left hand side of the model formula (preceding the tilde symbol), we specify the outcome variable. The right hand side consists of three parts, separated by vertical bars: The first part specifies the subgroup-specific fixed-effect model, which consists only of an intercept in this example; the second part specifies the random effects and the third part specifies the potential partitioning variables:

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

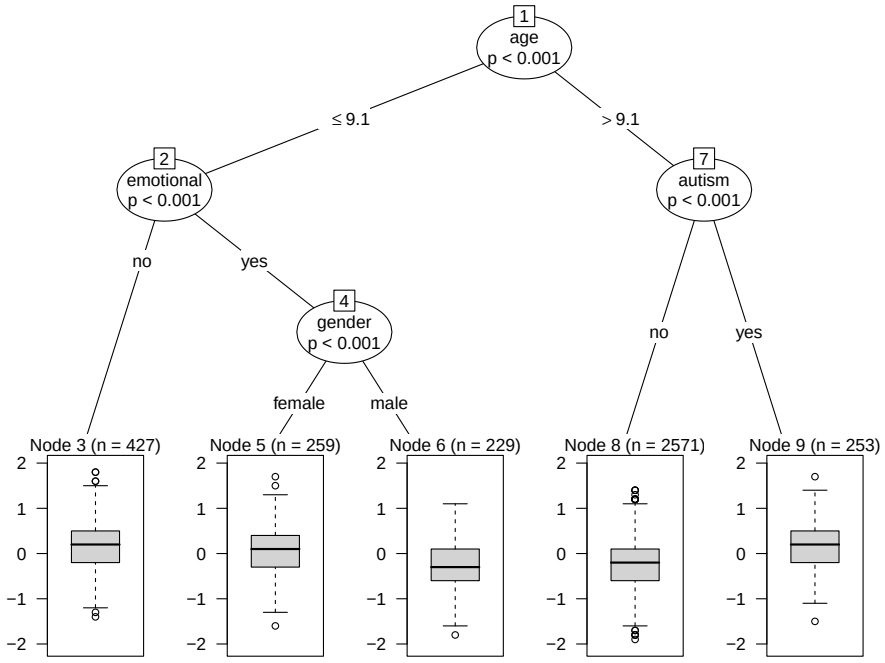

Figure 1: Linear mixed-effects model tree.

We specified only a single variable in the random-effects part, resulting in estimation of a random intercept with respect to `cluster_id`. More complex random effects can also be specified: for example, specifying the random-effects part as `(1 + age | cluster)` would yield a model with a random intercept as well as a random slope for age 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 a dichotomized response could be obained by:

```
R> glmmt <- glmertree(factor(outcome > 0) ~ 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

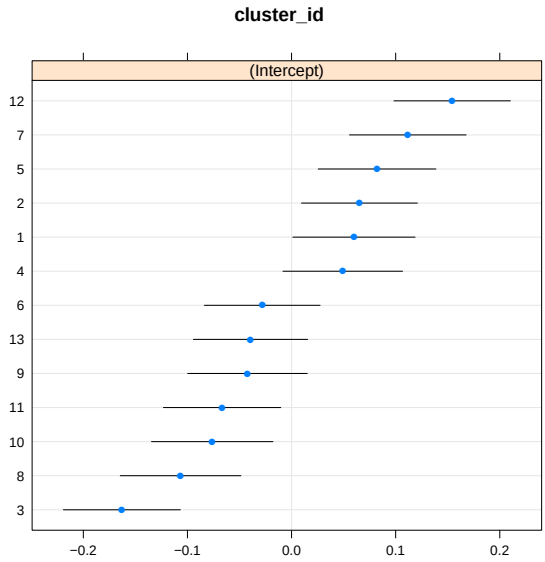

Figure 2: Random effects.

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 five subgroups: node 3 indicates that for patients with lower age and no emotional disorder, as well as for patients with higher age and autistic disorder, somewhat higher values for the response are observed. Thus, for these patients, poorer treatment outcomes are predicted. Somewhat better treatment outcomes are observed for those with lower age, an emotional disorder and female gender, and for those with higher age and no autistic disorder. The best treatment outcomes are observed among those with lower age, emotional disorder and male gender.

The predicted random effects are plotted in Figure 2. On average, patients at service provider 12 appear to have poorer outcomes, while patients at service provider 3 appear to have more favorable outcomes.

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

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

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

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

         1          2          3          4          5          6
0.08412602 0.25068474 0.25594172 0.15131909 0.34498973 0.30241507
```

```
         7          8          9         10
0.25353372 0.11452905 0.15131909 0.16260310
```

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 = MHserviceDemo[1:10, -7], re.form = NA)

         1          2          3          4          5          6
0.19073560 0.19073560 0.19073560 0.19073560 0.19073560 0.19073560
         7          8          9         10
0.09927958 0.19073560 0.19073560 0.19073560
```

2.1. 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(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 first plot does not indicate substantial variation in error variances across levels of the random effects. The second plot of fitted values against residuals also does not reveal a pattern indicating model misspecification.

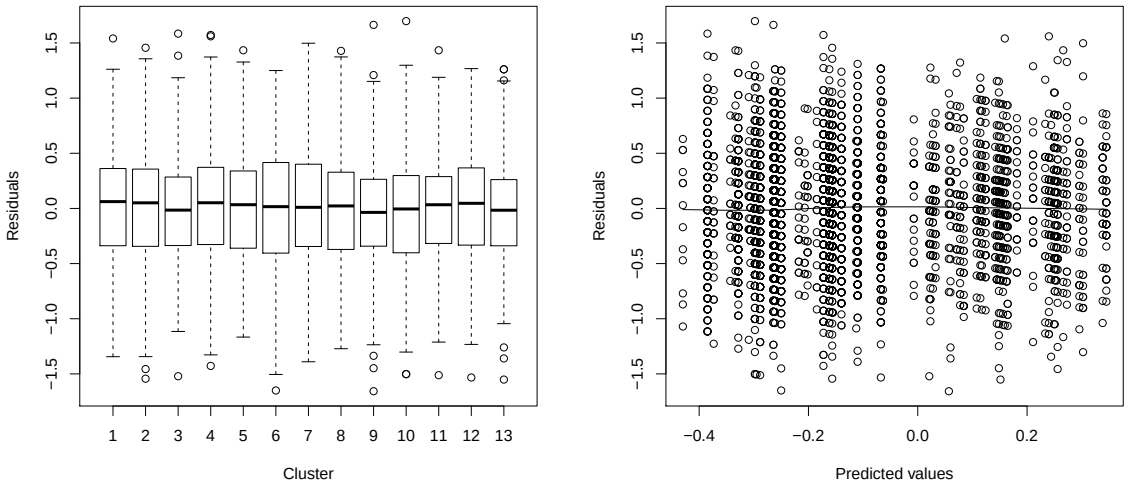

Figure 3: Residuals of the fitted linear mixed-effects model tree.

3. Detecting treatment-subgroup interactions in clustered data

The model in 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`).

With the left hand side of the model formula (preceding the tilde symbol), we specify the outcome variable. The right hand side consists of three parts, separated by vertical bars: The first part specifies the predictor variable(s) of the (generalized) linear model, the second part specifies the random effects and the third part specifies the potential partitioning variables:

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

Note that in the example above, the partitioning variables are continuous, but (ordered) categorical partitioning variables may also be specified. Also, we specified only 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 random-effects part as `(1 + age | cluster)` would yield a model with a random intercept as well as a random slope for age 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:

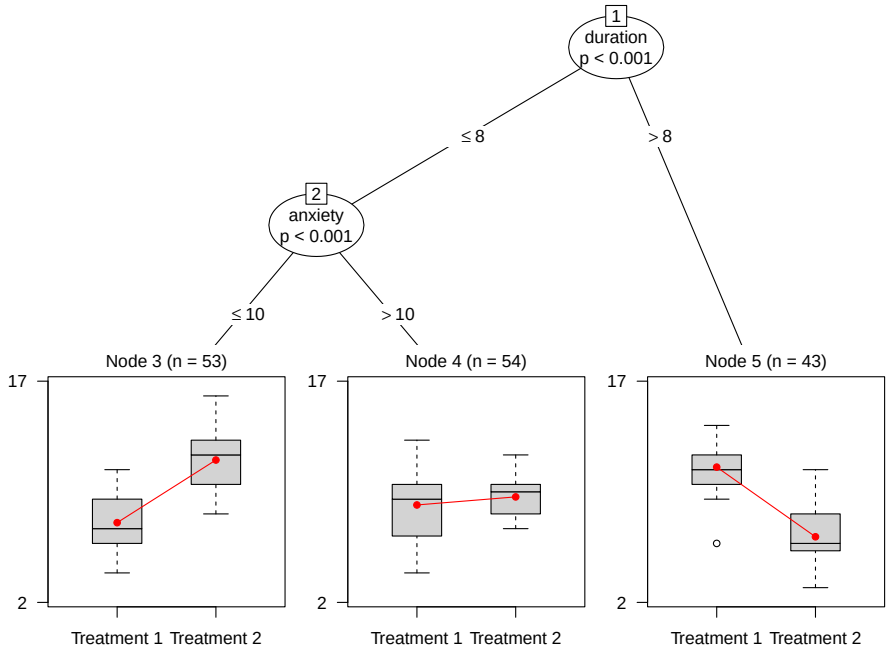

Figure 4: Linear mixed-effects model tree.

```
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 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.

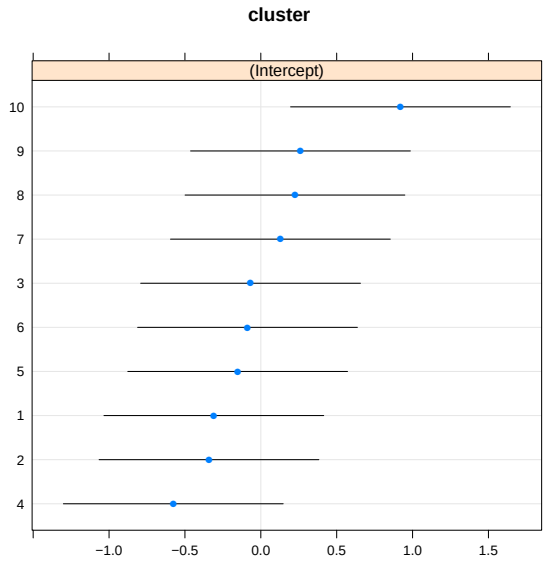

Figure 5: Random effects.

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` and `ranef` methods are available:

```
R> print(lmm_tree)

Linear mixed model tree

Model formula:
depression ~ treatment | age + duration + anxiety

Fitted party:
[1] root
|   [2] duration <= 8
|   |   [3] anxiety <= 10: n = 53
|   |                (Intercept) treatmentTreatment 2
|   |                   7.458519             4.183184
|   |   [4] anxiety > 10: n = 54
|   |                (Intercept) treatmentTreatment 2
|   |                   8.612009             0.513343
|   [5] duration > 8: n = 43
|                (Intercept) treatmentTreatment 2
|                  11.098602            -4.584979

Number of inner nodes:    2
Number of terminal nodes: 3
```

```
Number of parameters per node: 2
Objective function (residual sum of squares): 520.2838

Random effects:
$cluster
   (Intercept)
1  -0.30964507
2  -0.34154680
3  -0.06755161
4  -0.57675846
5  -0.15247332
6  -0.08761728
7   0.12905558
8   0.22500977
9   0.26125771
10  0.92026948

with conditional variances for "cluster"
```

```
R> coef(lmm_tree)

  (Intercept) treatmentTreatment 2
3    7.458519             4.183184
4    8.612009             0.513343
5   11.098602            -4.584979
```

```
R> ranef(lmm_tree)

$cluster
   (Intercept)
1  -0.30964507
2  -0.34154680
3  -0.06755161
4  -0.57675846
5  -0.15247332
6  -0.08761728
7   0.12905558
8   0.22500977
9   0.26125771
10  0.92026948

with conditional variances for "cluster"
```

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 first plot does not indicate substantial variation in error variances across levels of the random effects. The second plot of fitted values against residuals also does not reveal a pattern indicating model misspecification.

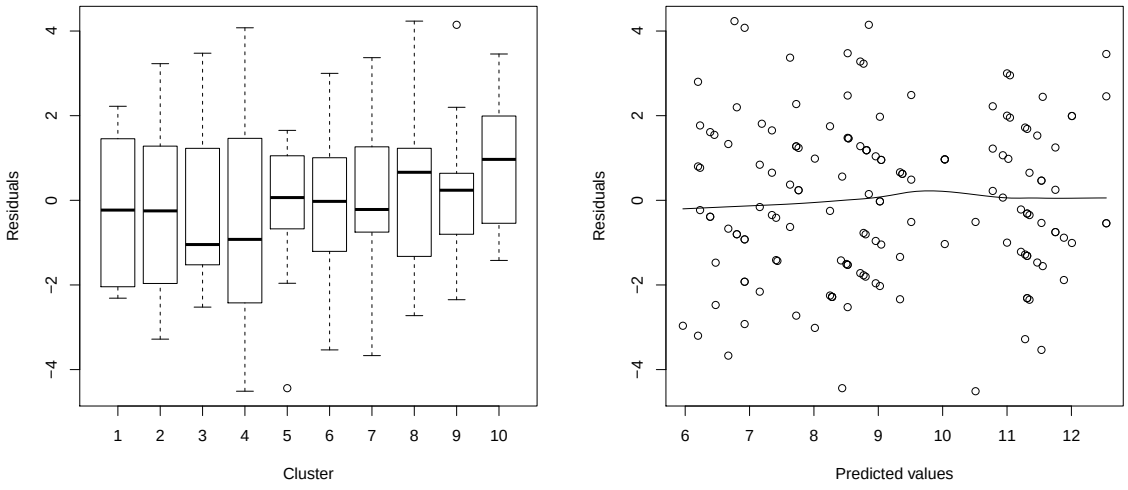

Figure 6: Residuals of the fitted linear mixed-effects model tree.

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/