POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

SpMEMs: an R package for semi-parametric mixed-effects models

Tesi di Laurea Magistrale in
Mathematical Engineering - Ingegneria Matematica

Author: **Davide Lo Piccolo**

Student ID: 970746
Advisor: Prof. Chiara Masci
Co-advisors: Prof. Francesca Ieva
Academic Year: 2022-23

Abstract

Mixed-effects models are a widely used branch of statistics nowadays and the literature focusing on their development is constantly evolving. In this thesis, we propose an R package for Semi parametric Mixed Effect Models, called SpMEMs, which has the task of collecting and automating into user-usable functions some of the latest results obtained in the research about mixed-effects models. In particular, we refer to two branches of research. The former concerns mixed-effects models for generalized responses where tree and random forest structures are assumed for the fixed-effects part. The latter concerns mixed-effects models for generalized, univariate, and biavariate responses where random effects assume a discrete distribution with an a priori unknown support. This particular innovation allows mixed-effects models to act as a method for unsupervised classification by joining similar grouping units into sub-populations. The thesis provides a theoretical explanation of the methods presented and investigates the R code in detail, by offering diversified examples useful for understanding the methods and related algorithms.

Keywords: R package, Semiparametric methods, Mixed effects models, Regression and classification trees, random forest

Abstract in lingua italiana

I modelli a effetti misti sono una branca della statistica molto utilizzata al giorno d'oggi e la letteratura che si concentra sul loro sviluppo è in continua evoluzione. In questa tesi, proponiamo un pacchetto R per modelli a effetti misti semiparametrici, chiamato SpMEMs, che ha il compito di raccogliere e automatizzare in funzioni utilizzabili dall'utente alcuni degli ultimi risultati ottenuti nell'ambito della ricerca per questo tipo di modelli. In particolare si fa riferimento a due filoni di ricerca, il primo riguarda modelli a effetti misti per risposte generalizzate dove per la parte a effetti fissi vengono assunte strutture ad albero e random forest; il secondo riguarda modelli a effetti misti per risposte generalizzate, univarite e biavariate in cui gli effetti random assumono una distribuzione discreta con supporto sconosciuto a priori. Questa particolare innovazione permette ai modelli a effetti misti di agire come un metodo per la classificazione non supervisionata unendo grupppi simili di unità statistiche in sotto-popolazioni. La tesi fornisce una spiegazione teorica dei metodi presentati, e investiga nel dettaglio la parte di codice necessaria all'utente per l'utilizzo del pacchetto offrendo vari esempi utili alla comprensione dei metodi e degli algoritmi relativi.

Parole chiave: Pacchetto R, Metodi Sempi parametricis,Modelli a effetti misti, Alberi di regressione e classificazione, Foresta casuale

Contents

1 | Introduction

Mixed effects models, also known as multilevel models or hierarchical models, have become increasingly popular in various fields of research due to their ability to handle complex data structures [25]. These models are particularly useful when analyzing data with multiple levels of variation, such as repeated measurements on individuals or data collected from different sites. When dealing with hierarchical data, compared to traditional methods, mixed effects models allow for accurate estimation of fixed and random effects, provide more precise predictions, and account for dependencies among observations. This brief introduction highlights the potential of mixed effects models in handling complex data structures. However, a more in-depth investigation of these models can reveal even greater power and versatility, enabling researchers to analyze increasingly challenging datasets. In the R package we present in this thesis we implement and show the latest advancement of mixed effect models to study different types of hierarchical data. The material of the package derive from two strands of research. The first one presented in [6] and [24] investigate the idea of joining a generalized mixed models with a tree based methods. As we already said Multilevel models are designed to account for the hierarchical structure of data, however, Generalized Linear Mixed Models (GLMM) assume a linear relationship between the covariates and the expected value of the response variable. On the other hand, tree-based methods like the classification and regression tree (CART) [15] or random forest [4] models identify dominant patterns in the training data to establish the relationship between the response and predictor variables. In particular, the first research stream consists in the development of tree-based mixed-effects models under two different setting. In the first case, a tree structure is assumed for the fixed-effects part [6], while in the second case, a Random Forest (RF) is assumed [24]. This approach allows to create a novel method, for a non-Gaussian response variable, which is able to preserve the flexibility of the CART and the random forest models and to extend it to a clustered data structure, where multiple observations can be viewed as being sampled within groups. In the literature, similar methods have been proposed in this context. The first one, called Mixed Effects Regression Tree (MERT) [8] and it substitutes the linear combination of the covariates in the fixed effects part of a LMM with a regression tree,

built with the same set of covariates. In [31], the authors present an analogous method, but with a different estimation procedure, called Random Effects Expectation Maximization tree (RE-EM tree), that deals with both multilevel and longitudinal data. With the aim of improving the accuracy in predictions, regression trees are replaced by a RF in [9], where the authors develop a method called Mixed Effects Random Forest (MERF). All these methods deal with a Gaussian response variable and they are not suitable for classification problems. Nonetheless, some developments for different types of response variables have also been done. In [10], the MERT approach is extended to non-gaussian data and a Generalized Mixed Effects Regression Tree (GMERT) is proposed. This algorithm is basically the Penalized Quasi Likelihood (PQL) algorithm used to fit Generalized Linear Mixed Models (GLMMs) where the weighted linear mixed effect pseudo-model is replaced by a weighted MERT pseudo-model. In [5], the authors propose a generalized linear mixed-effects model tree (GLMER tree) algorithm, that alternates the estimates of a GLM tree and a mixed-effects model until convergence. Lastly, the most recent work is proposed in [32], where the authors develop a decision tree method for modelling clustered and longitudinal binary outcomes using a Bayesian setting.

The second research stream shifts attention to the random component of the model. In fact, in articles [16], [17], [18] and [19] a change is proposed regarding the distribution of random parameters, which is no longer intended to be treated as a continuous distribution but a discrete one with an unknown finite number of mass point. The advantage of this approach is that it allows the mixed-effects model to be used both as a predictive model and as an unsupervised classification method. In fact, it is possible to reduce the support of the discrete distribution by detecting a latent structure among the higher level of hierarchy. This type of approach is conjugated in 3 case histories that are identified by the shape of the response variable, specifically Semi-Parametric Estimation Maximization (SPEM) [16] deals with the case where the response variable is continuous and univariate, Bivariate Semi-Parametric Estimation Maximization (BSPEM) [17] deals wit the case where the response variable is continuous and bivariate, and finally, Multinomial Semi-Parametric Estimation Maximization and Joint Multinomial Semi-Parametric Estimation Maximization ([18] and [19] respectively) that deal with the case where the response is univariate multinomial.

In the methodological literature, two lines of research about the identification of subpopulations are Growth Mixture Models (GMM) [21], [22] and Latent Class Mixture Models [20], [23], [34]. Conventional growth modelling is applied to longitudinal data and it is used to estimate the average growth, the amount of variation across individuals in growth intercept and slopes and the influence of covariates on this variation. It can be described

as a random effect model where intercept and slope vary across individuals. However, conventional growth models assume that individuals come from a single population and that a single growth trajectory can approximate the entire population. Growth mixture models relax this assumption and assume that there are differences in growth parameters across unobserved subpopulations. They allow for the existence of latent trajectory classes where different groups of individual growth trajectories vary around different behaviors. In other words, the average association between covariates and the outcome varies across latent classes and also, within classes, individuals also vary randomly in their coefficients. The results are separate growth models for each latent class. Latent Class Growth Analysis (LCGA) is a special case of GMM where the variance and covariance parameters are assumed to be zero, implying that all the individuals within a latent class are homogeneous. Individuals within a latent class are assumed to have identical random effects. Conceptually, these methods are very similar to the one that we propose, especially the special case of LCGA, since we also assume that individuals within latent classes have identical random effects. Nonetheless, there are two main differences between our approach and the one of GMM and LCGA. First GMM/LCGA are thought for modelling longitudinal changes and not variation within groups. Second GMM/LCGA need to fix a priori the number of latent classes, while our approach estimates it together with the other unknown parameters. There are numerous extensions and applications of GMM, [12], [26], but none of them includes the estimation of the number of latent classes. Indeed, these past methods require that the analysts estimate a series of models, where each model assumes a different number of clusters, and then use model fit statistics to compare these models to select the best fitting model. In our approach, the analyst specifies a caliper, the maximum distance between two clusters such that the two can be collapsed, and the algorithm then estimates the number of clusters. Latent class mixture models are even more related to our approach since they consider linear mixed models where the assumption of normality of random effects is relaxed. They also assume a discrete distribution for the random effect coefficients and they are used to uncover distinct subpopulations (latent classes) and to classify individuals. But also this approach requires a fixed number of latent classes, chosen a priori. In the framework of latent structure analysis, an other branch of research related to ours is the one about Latent Trait Analysis (LTA) [3], LTA, also called Item Response Theory (IRT), is used for the analysis of categorical data. It performs the reduction of a set of binary or ordered-category variables into a smaller set of classes and it is used both to calibrate items and to derive latent trait estimates that are then used in subsequent analysis. The common aspect of this method with the ones described above and, at the same time, the main difference with our method is, again, the fact that they need to fix a priori the number of latent subpopolations. The choice

of the number of latent classes (mass points) is not trivial when the sample is very big or the knowledge about possible different trends across the individuals (groups) is limited. Our case study represents a clear example of a sample composed by hundreds of groups, within which we do not know how many different subpopulations exist. For this reason, in the perspective of performing dimensionality reduction without any assumption about the final dimension, estimates, together with the other parameters, also the number of existing subpopulations, a significant value-added with respect to the existing literature.

1.0.1. Overview of the paper

The paper is structured as follows:

- In Chapter 2, we provide a more in-depth introduction to the previously mentioned methods, with a focus on the algorithmic composition and the theoretical definition of the models.
- In Chapter 3 we get into the core of the explanation of the package. All the main functions related to the aforementioned statistical methods are explained in detail with emphasis on each of the inputs and outputs that comprise it. Next, all the auxiliary functions related to each method that make up a very important tool for understanding the outputs are listed.
- In Chapter 4 of the thesis presents some cases through which an example of the use of various package functions can be seen. In the first part, the process of generating data is explained, while in the second we show the applications of the functions to the toy dataset.

All chapters of the thesis are dissected into five parts one for each statistical model.

2 | Methodological details

2.1. Generalised Mixed-Effects tree-based model

Consider a general GLMM linear predictor in this model, which is an extension of a generalized linear model [1], includes both fixed and random effects. For a GLMM with a two-level hierarchy, each observation j, for $j = 1, \ldots, n_i$, is nested within a group i, for $i = 1, \ldots,$ N. Let $\boldsymbol{y}_i = (y_{1i}, \ldots, y_{n_i i})$ be the n_i-dimensional response vector for observations in the i-th group. Conditionally on random effects denoted by $\boldsymbol{b}_i$, a GLMM assumes that the elements of $\mathbf{y}_i$ are independent, with density function from the exponential family, of the form

$$f_i\left(y_{ij} \mid \boldsymbol{b}_i\right) = \exp\left[\frac{y_{ij}\eta_{ij} - a\left(\eta_{ij}\right)}{\phi} + c\left(y_{ij}, \phi\right)\right]$$

where $a(\cdot)$ and $c(\cdot)$ are specified functions, η_{ij} is the natural parameter and ϕ is the dispersion parameter. In addition, we have

$$\begin{aligned} E\left[y_{ij} \mid \boldsymbol{b}_i\right] &= a'\left(\eta_{ij}\right) = \mu_{ij} \\ \operatorname{Var}\left[y_{ij} \mid \boldsymbol{b}_i\right] &= \phi a''\left(\eta_{ij}\right) \end{aligned}$$

A monotonic, differentiable link function $g(\cdot)$ specifies the function of the mean that the model equates with the systematic component. Usually, the canonical link function is used, i.e., $g = a'^{-1}$. From now on, without loss of generality, the canonical link function is used. In this case, the model is the following:

$$\begin{aligned} \boldsymbol{\mu}_i &= E\left[\boldsymbol{Y}_i \mid \boldsymbol{b}_i\right] \quad i = 1, \ldots, N \\ g\left(\boldsymbol{\mu}_i\right) &= \boldsymbol{\eta}_i \\ \boldsymbol{\eta}_i &= X_i\boldsymbol{\beta} + Z_i\boldsymbol{b}_i \\ \boldsymbol{b}_i &\sim N_q(\mathbf{0}, \Psi) \quad \text{ind.} \end{aligned} \tag{2.1}$$

where i is the group index, N is the total number of groups, n_i is the number of observations within the i-th group and $\sum_{i=1}^{N} n_i = J$. η_i is the n_i-dimensional linear predictor vector. In addition, X_i is the $n_i \times (p+1)$ matrix of fixed-effects regressors of observations in group i, $\boldsymbol{\beta}$ is the $(p+1)$-dimensional vector of their coefficients, Z_i is the $n_i \times q$ matrix of regressors for the random effects, $\boldsymbol{b}_i$ is the $(q+1)$-dimensional vector of their coefficients and Ψ is the $q \times q$ within-group covariance matrix of the random effects. Fixed effects are identified by parameters associated with the entire population, whereas random ones are identified by group-specific parameters.

The innovation in the two methods proposed in [6] and [24], is to extend the application of tree-based mixed models to other classes of exponentially distributed response variables.. The two methods share the matrix formulation of the model that is now presented. The response variable Y from an exponential family distribution makes up the model's random element. The fixed effect part is replaced by the function $f(X_i)$, which is evaluated using a tree-based approach, as opposed to being linear as in Eq.(1). Thus, the matrix formulation of the model is the following:

$$\begin{aligned} \boldsymbol{\mu}_i &= E[\boldsymbol{Y}_i \mid \boldsymbol{b}_i] \quad i = 1, \ldots, N \\ g(\boldsymbol{\mu}_i) &= \boldsymbol{\eta}_i \\ \boldsymbol{\eta}_i &= \boldsymbol{f}(X_i) + Z_i \boldsymbol{b}_i \\ \boldsymbol{b}_i &\sim N_q(\mathbf{0}, \Psi) \quad \text{ind.} \end{aligned} \tag{2.2}$$

As in a GLMM, $\boldsymbol{b}_i$ and $\boldsymbol{b}_{i'}$ are independent for $i \neq i'$. The difference between the two methods consist in the estimation of the fixed effect, in particular the the difference relies on the type of tree based method they use. GMET (Generalised Mixed-Effects Tree) model estimates it using a non-parametric CART model applied to the entire population, while the GMERF (Generalised Mixed-Effects Random Forest) uses a random forest model.

2.1.1. Generalised Mixed-Effects Tree

Let's consider the model in the Eq.(2.2), the function $\boldsymbol{f}(X_i)$ representing the fixed effect part is estimated using a non-parametric CART tree model applied to the entire population, whilst group-specific parameters are used to identify random effects.

The fundamental goal of the algorithm, which is similar to the RE-EM tree's methodology [30], is to separate the estimation of fixed effects from random effects. However the GMET process is not iterative. The structure of the algorithm is the following:

1. Initialise the estimated random effects $\boldsymbol{b}_i$ to zero.

2. Estimate the target variable μ_{ij} through a generalised linear model (GLM), given fixed-effects covariates $x_{ij} = (x_{ij1}, \ldots, x_{ijp})^T$ for $i = 1, \ldots, N$ and $j = 1, \ldots, n_i$. Get estimate $\hat{\mu}_{ij}$ of target variable μ_{ij}.

3. Build a regression tree approximating f using $\hat{\mu}_{ij}$ as dependent variable and $x_{ij} = (x_{ij1}, \ldots, x_{ijp})^T$ as vector of covariates. This regression tree identifies a number L of terminal nodes R_ℓ, for $\ell = 1, \ldots, L$, and each observation ij, described by its set of covariates x_{ij}, belongs to one of the terminal nodes. Through this regression tree, we define a set of indicator variables $I(x_{ij} \in R_\ell)$, for $\ell = 1, \ldots, L$, where $I(x_{ij} \in R_\ell)$ takes value 1 if observation ij belongs to the ℓ-th terminal node and 0 otherwise.

4. Fit the mixed effects model in Eq.(2), using y_{ij} as a response variable and the set of indicator variables $I(x_{ij} \in R_\ell)$ as fixed-effects covariates (dummy variables). Specifically, for $i = 1, \ldots, N$ and $j = 1, \ldots, n_i$, we have $g(\mu_{ij}) = I(\boldsymbol{x}_{ij} \in R_\ell)\gamma_\ell + \boldsymbol{z}_{ij}^T \boldsymbol{b}_i$. Extract $\hat{\boldsymbol{b}}_i$ from the estimated model.

5. Replace the predicted response at each terminal node R_ℓ of the tree with the estimated predicted response $g(\hat{\gamma}_\ell)$ from the mixed-effects model fitted in step 4 .

2.1.2. Generalised Mixed-Effects Random Forest

The GMERF model estimates the function $\boldsymbol{f}(X_i)$ of the model presented in Eq.(2.2) with a random forest. To implement GMERF model, we need to decouple the estimation of fixed and random effects parts, alternating them until convergence. To this purpose, we note that, if random effects were known, the GMERF model implies that we could fit a RF to estimate f using $\eta_{ij} - \mathbf{Z}_{ij}^T \mathbf{b}_i$ as dependent variable. Similarly, if the population-level effects f were known, then we could estimate the random effects using a traditional mixed-effects linear model with response corresponding to $\eta_{ij} - f(\mathbf{X}_{ij})$. As neither the random effects nor the fixed effects are known, we implement an iterative method that alternates, until convergence, the estimation of the RF, relative to the fixed effects part, with the estimation of the random effects. The convergence is reached when the difference between the random effects estimates at two consecutive iterations is lower than a fixed tolerance. A second important aspect to be faced is that η_i is not known and it cannot be directly deduced from data. In line with the one proposed in [7], is estimating it by means of a standard GLM model using as covariates the fixed effects covariates. The pseudo-code of the estimation procedure is shown in Algorithm 1. To predict a new observation $[\mathbf{x}_{ij}; \mathbf{z}_{ij}]$ we use the formula:

$$\hat{\eta}_{ij} = \hat{f}(\mathbf{x}_{ij}) + \mathbf{z}_{ij}^T \hat{b}_i \tag{2.3}$$

where $\hat{f}$ is the RF estimated by the algorithm and $\mathbf{b}_i$ is the vector of the random effects coefficients related to the i th group. The prediction of $\widehat{\mu}_{ij}$ is obtained by applying to the corresponding $\hat{\eta}_{ij}$ the inverse link function g^{-1}.

Algorithm 2.1 GMERF model estimation procedure

Input:

y- vector with responses y_{ij}
cov- data frame with all covariates
gr- vector with the grouping variable for each observation
$znam$- vector with names of covariates to be used as random effects
$xnam$- vector with names of covariates to be used as fixed effects
fam- distribution of y (must be part of the exponential family)
b_0- optional matrix of initial values for each $\mathbf{b}_i$
$toll$ threshold to decide whether our estimation converged or not
$itmax$ maximum number of iterations

$Z \leftarrow (1; cov[znam])$ {to include also the random intercept}
Initialize b to a matrix of zero (if b_0 is not given) {Each column $b[i,]$ of b will be the i-th random coefficients $\mathbf{b}_i$}
$all.b[0] = b$
fit a GLM model using y as response and cov as matrix of covariates
$eta \leftarrow$ estimated η_{ij} by the GLM model
$it \leftarrow 1$
while $it < itmax$ **and not** $conv$ **do**
  $targ \leftarrow eta - Z \times b$
  fit a random forest model using $targ$ as target and cov as predictor matrix
  $fx \leftarrow$ fitted values of the forest model
  fit the GLMM $\eta_{ij} - f(\mathbf{x}_{ij}) = \mathbf{z}_{ij}^T \times \mathbf{b}_i$
  $all.b[it] \leftarrow b \leftarrow$ the estimated b from the model
  $M \leftarrow \max(abs(b - all.b[it-1]))$
  $(i, j) \leftarrow \text{argmax}(abs(b - all.b[it-1]))$
  $tr \leftarrow M/all.b[it-1](i, j)$
  if $tr < toll$ **then**
    $conv \leftarrow$ **true**
  else
    $conv \leftarrow$ **false**
  end if
  $it++$
end while
if not $conv$ **then**
  give a warning
end if
Output:

the final GLMM fitted
the final forest model fitted
b, the final estimation of the random coefficients
it, the number of iterations

2.2. SemiParametric Mixed Effects Models For Unsupervised Classification

Consider a general mixed-effects (two-level) linear model, where each observation j, for $j = 1, \dots, n_i$, is nested within a group i, for $i = 1, \dots, N$. The model takes the following form:

$$\begin{aligned} \mathbf{y}_i &= \mathbf{X}_i\boldsymbol{\beta} + \mathbf{Z}_i\mathbf{b}_i + \boldsymbol{\epsilon}_i \qquad i = 1, \dots, N \\ \boldsymbol{\epsilon}_i &\sim \mathcal{N}(\mathbf{0}, \sigma^2 \mathbb{1}_{n_i}) \qquad ind. \end{aligned} \tag{2.4}$$

where i is the group index, N is the total number of groups, n_i is the number of observations within the i-th group and $\sum_{i=1}^{N} n_i = J$. $\mathbf{y}_i = (y_{1i}, \dots, y_{n_i i})$ is the n_i-dimensional vector of response variable within the $i-$th group, $\mathbf{X}_i$ is the $n_i \times (p+1)$ matrix of covariates having fixed effects, $\boldsymbol{\beta}$ is the (p+1)-dimensional vector of fixed coefficients, $\mathbf{Z}_i$ is the $n_i \times (r+1)$ matrix of covariates having random effects, $\mathbf{b}$ is the (r+1)-dimensional vector of random coefficients and $\boldsymbol{\epsilon}_i$ is the vector of errors.

In the parametric framework of mixed-effects linear models, random coefficients are assumed to be distributed according to a Normal distribution with unknown parameters that, together with the coefficients of fixed effects and σ^2, can be estimated through methods based on the maximization of the likelihood or the restricted likelihood functions [25].

The main novelty first introduced in [16] and then further developed in the next two articles([17],[18]) of the second branch of research is that we move to a semi-parametric framework, assuming the coefficients $\mathbf{b}_i$ to be distributed according to a discrete distribution $\mathcal{P}^*$, assuming M sets of values $(c_{0l}, \dots, c_{rl})$ for $l = 1, \dots, M$, where $M \leq N$ and it is unknown a priori. This means that each group i, for $i = 1, \dots, N$, is assigned to a subpopulation l, that is characterized by random parameters $(c_{0l}, \dots, c_{rl})$. This semi-parametric modelling enables to identify a latent structure among the groups, that are clustered by the model into an unknown number of discrete masses. Therefore, the two main advantages are that, first of all, we can identify how many latent subpopulations exist within the groups of data and, second, we can estimate the parameters associated to each subpopulation, pointing out their differences.

Under these assumptions, the semi-parametric mixed-effects model takes the following form:

$$\begin{aligned} \mathbf{y}_i = \mathbf{X}_i\boldsymbol{\beta} + \mathbf{Z}_i\mathbf{c}_l + \boldsymbol{\epsilon}_i \qquad i = 1, \dots, N \qquad l = 1, \dots, M \\ \boldsymbol{\epsilon}_i \sim \mathcal{N}(\mathbf{0}, \sigma^2 \mathbb{1}_{n_i}) \qquad ind. \end{aligned} \tag{2.5}$$

This formulation is the general formulation to describe all the model of the second strand of research. The differences between the methods presented is in the response that vary both in dimension and type. We present the case in which the response variable is univariate considering one random effect and one fixed effect. While the latter situation just presents the formulation model, the former case is thoroughly investigated. Since the algorithms are a generalization of the SPEM (Semi Parametric Estimation Maximixation) method, they are taught through the sketches.

The formulation of this first case is:

$$\begin{aligned} \mathbf{y}_i = \mathbf{x}_i\beta + \mathbf{1}c_{0l} + \mathbf{z}_i c_{1l} + \boldsymbol{\epsilon}_i \qquad i = 1, \dots, N \qquad l = 1, \dots, M \\ \boldsymbol{\epsilon}_i \sim \mathcal{N}(\mathbf{0}, \sigma^2 \mathbb{1}_{n_i}) \qquad ind. \end{aligned} \tag{2.6}$$

where $\mathbf{1}$ is the n_i-dimensional vector of 1, $M \leq N$ is the number of subpopulations (mass points) unknown a priori. Coefficients $\mathbf{c}_l$, for $l = 1, \dots, M$, are distributed according to a probability measure $\mathcal{P}^*$ that belongs to the class of all probability measures on $\mathbb{R}^2$. $\mathcal{P}^*$ is a discrete measure with M support points that can then be interpreted as the mixing distribution that generates the density of the stochastic model in Eq. (2.6). The ML estimator $\hat{\mathcal{P}}^*$ of $\mathcal{P}^*$ is obtained following the theory of mixture likelihoods in [13] and [14]. So, the ML estimator of the random effects distribution can be expressed as a set of points $(\mathbf{c}_1, \dots, \mathbf{c}_M)$, where $M \leq N$ and $\mathbf{c}_l \in \mathbb{R}^2$ for $l = 1, \dots, M$, and a set of weights $(w_1, \dots, w_M)$, where $\sum_{l=1}^{M} w_l = 1$ and $w_l \geq 0$ for each $l = 1, \dots, M$.

Given this, the algorithm proposed for the joint estimation of σ^2, β, $(\mathbf{c}_1, \dots, \mathbf{c}_M)$ and $(w_1, \dots, w_M)$, that is performed through the maximization of the likelihood, mixture by the discrete distribution of the random effects,

$$\begin{aligned} L(\boldsymbol{\beta}, \sigma^2|\mathbf{y}) = p(\mathbf{y}|\boldsymbol{\beta}, \sigma^2) = \\ \sum_{l=1}^{M} \frac{w_l}{(2\pi\sigma^2)^{\frac{J}{2}}} exp\left\{ -\frac{1}{2\sigma^2} \sum_{i=1}^{N} \sum_{j=1}^{n_i} (y_{ij} - \beta x_{ij} - c_{0l} - c_{1l} z_{ij})^2 \right\}, \end{aligned} \tag{2.7}$$

with respect to the fixed coefficient β, the error variance σ^2 and the random effects distribution $(\mathbf{c}_l, w_l)$, for $l = 1, \dots, M$. For each $l = 1, \dots, M$, $\mathbf{c}_l$ represents the group-specific parameters and w_l the corresponding weight.

2.2.1. Semi-parametric mixed-effects models for a continuous response

The proposed EM algorithm is an iterative algorithm that alternates two steps: the expectation step (E step) in which we compute the conditional expectation of the likelihood function with respect to the random effects, given the observations and the parameters computed in the previous iteration; and the maximization step (M step) in which we maximize the conditional expectation of the likelihood function. The observations are the values of the answer variable y_{ij} and of the covariates z_{ij} and x_{ij}, for $j = 1, \ldots, n_i$ and $i = 1, \ldots, N$. The parameters to be estimated are the random coefficients $\mathbf{c}_l$ with their weights w_l, for $l = 1, \ldots, M$, the fixed coefficient β and the variance σ^2. The algorithm allows the number n_i, for $i = 1, \ldots, N$, of observations to be different across groups, but, within each group missing data are not handled, i.e. missing values of y, z and x for the n_i units are not allowed. At each iteration, the EM algorithm updates the parameters in order to increase the likelihood in (2.7) and it continues until the convergence or until a fixed number of iterations (`it`) is reached. In particular, the update is given by:

$$w_l^{(up)} = \frac{1}{N} \sum_{i=1}^{N} W_{il} \qquad \text{for } l = 1, \ldots, M \tag{2.8}$$

$$(\beta^{(up)}, \mathbf{c}_1^{(up)}, \ldots, \mathbf{c}_M^{(up)}, \sigma^{2(up)}) = \underset{\beta, \mathbf{c}_l, \sigma^2}{\arg\max} \sum_{l=1}^{M} \sum_{i=1}^{N} W_{il} \, \ln p(\mathbf{y}_i | \beta, \sigma^2, \mathbf{c}_l) \tag{2.9}$$

where

$$W_{il} = \frac{w_l \, p(\mathbf{y}_i | \beta, \sigma^2, \mathbf{c}_l)}{\sum_{k=1}^{M} w_k \, p(\mathbf{y}_i | \beta, \sigma^2, \mathbf{c}_k)} \tag{2.10}$$

and

$$p(\mathbf{y}_i | \beta, \sigma^2, \mathbf{c}) = \frac{1}{(2\pi\sigma^2)^{\frac{n_i}{2}}} exp \left\{ -\frac{1}{2\sigma^2} \sum_{j=1}^{n_i} (y_{ij} - \beta x_{ij} - c_{0l} - c_{1l} z_{ij})^2 \right\}. \tag{2.11}$$

The weight $w_l^{(up)}$ is the mean over the N groups of their weights related to the $l-$th subpopulation. Coefficients W_{il} represent the probability of $\mathbf{b}_i$ being equal to $\mathbf{c}_l$ conditionally to observations $\mathbf{y}_i$ and given the fixed coefficient β and the variance σ^2. Refer to [16] for further details on the derivation of the process.

The maximization (M step) in equation (2.9) involves two steps and it is done iteratively. In the first step, we compute the *arg-max* with respect to the support points $\mathbf{c}_l$, keeping β and σ^2 fixed to the last computed values. In this way, we can maximize the expected log-likelihood (computed in the E step) with respect to all support points $\mathbf{c}_l$ separately, that means

$$\mathbf{c}_l^{(up)} = \arg\max_{\mathbf{c}} \sum_{i=1}^{N} W_{il} \ln p(\mathbf{y}_i|\beta, \sigma^2, \mathbf{c}) \qquad l = 1, \ldots, M. \tag{2.12}$$

Since we are considering the linear case, it is possible to perform this maximization step in closed-form. With regard to model (2.6), the estimates of the random effects are obtained by means of the weighted least squares method and are the following:

$$\hat{c}_{0l} = \frac{\sum_{i=1}^{N} w_{il} \sum_{j=1}^{n_i} (y_{ij} - \hat{\beta} x_{ij} - \hat{c}_{1l} z_{ij})}{\sum_{i=1}^{N} w_{il} n_i} \tag{2.13}$$

and

$$\begin{aligned} \hat{c}_{1l} = & \frac{\sum_{i=1}^{N} w_{il} \sum_{j=1}^{n_i} y_{ij} z_{ij} - \frac{(\sum_{i=1}^{N} w_{il} \sum_{j=1}^{n_i} y_{ij})(\sum_{i=1}^{N} w_{il} \sum_{j=1}^{n_i} z_{ij})}{n_i \sum_{i=1}^{N} w_{il}}}{\sum_{i=1}^{N} w_{il} \sum_{j=1}^{n_i} z_{ij}^2 - \frac{(\sum_{i=1}^{N} w_{il} \sum_{j=1}^{n_i} z_{ij})^2}{\sum_{i=1}^{N} w_{il} n_i}} \\ & + \frac{\frac{\hat{\beta}(\sum_{i=1}^{N} w_{il} \sum_{j=1}^{n_i} z_{ij})(\sum_{i=1}^{N} w_{il} \sum_{j=1}^{n_i} x_{ij})}{n_i \sum_{i=1}^{N} w_{il}} - \hat{\beta} \sum_{i=1}^{N} w_{il} \sum_{j=1}^{n_i} x_{ij} z_{ij}}{\sum_{i=1}^{N} w_{il} \sum_{j=1}^{n_i} z_{ij}^2 - \frac{(\sum_{i=1}^{N} w_{il} \sum_{j=1}^{n_i} z_{ij})^2}{\sum_{i=1}^{N} w_{il} n_i}}. \end{aligned} \tag{2.14}$$

In the second step, we fix the support points of the random effects distribution computed in the previous step and we compute the *arg-max* in equation (2.9) with respect to β and σ^2. Again, this step can be done in closed-form and the estimates of the parameters, with regard to model (2.6), obtained by means of the weighted least squares method, are:

$$\hat{\beta} = \frac{\sum_{l=1}^{M} \sum_{i=1}^{N} w_{il} \sum_{j=1}^{n_i} (y_{ij} x_{ij} - \hat{c}_{0l} x_{ij} - \hat{c}_{1l} z_{ij} x_{ij})}{\sum_{l=1}^{M} \sum_{i=1}^{N} w_{il} \sum_{j=1}^{n_i} x_{ij}^2} \tag{2.15}$$

and

$$\hat{\sigma}^2 = \frac{\sum_{l=1}^{M} \sum_{i=1}^{N} w_{il} \sum_{j=1}^{n_i} (y_{ij} - \hat{\beta} x_{ij} - \hat{c}_{0l} - \hat{c}_{1l} z_{ij})^2}{n_i \sum_{l=1}^{M} \sum_{i=1}^{N} w_{il}}. \tag{2.16}$$

Notice that, since $w_l = p(\mathbf{b}_i = \mathbf{c}_l)$, then

$$W_{il} = \frac{w_l\, p(\mathbf{y}_i|\beta, \sigma^2, \mathbf{c}_l)}{\sum_{k=1}^{M} w_k\, p(\mathbf{y}_i|\beta, \sigma^2, \mathbf{c}_k)} = \frac{p(\mathbf{b}_i = \mathbf{c}_l)\, p(\mathbf{y}_i|\beta, \sigma^2, \mathbf{c}_l)}{p(\mathbf{y}_i|\beta, \sigma^2)} = \\ = \frac{p(\mathbf{y}_i, \mathbf{b}_i = \mathbf{c}_l|\beta, \sigma^2)}{p(\mathbf{y}_i|\beta, \sigma^2)} = p(\mathbf{b}_i = \mathbf{c}_l|\mathbf{y}_i, \beta, \sigma^2). \tag{2.17}$$

Therefore, in order to compute the point $\mathbf{c}_l$ for each group i, for $i = 1, \ldots, N$, we maximize the conditional probability of $\mathbf{b}_i$ given the observations $\mathbf{y}_i$, the coefficient β and the error variance σ^2. The estimation of the coefficients $\mathbf{b}_i$ of the random effects for each group is obtained maximizing W_{il} over l, that is

$$\hat{\mathbf{b}}_i = \mathbf{c}_{\tilde{l}} \quad \text{where} \quad \tilde{l} = \arg\max_{l} W_{il} \quad i = 1, \ldots, N. \tag{2.18}$$

As anticipated before, the initialization of the support points is done in a robust and generalizable way. The algorithm starts considering N support points for the coefficients of random effects and a starting estimate for the coefficients of fixed effects. In particular, the initialization of all these parameters is done in the following way:

- random effects: the starting N support points are obtained fitting a simple linear regression within each group and estimating the couple of parameters (both the intercept and the slope) for each one of the N groups. The weights are uniformly distributed on these N support points;
- fixed effects: the starting values of β and σ^2 are estimated by fitting a unique linear regression on the entire population (without distinction among the groups).

Nonetheless, if the number of starting support points N is extremely large, the algorithm is relatively slow and using N starting support points becomes not strictly necessary. In this case, the initialization of the support points of the random effect distribution is done in the following way:

- we choose a number N* $<$ N of support points;
- we extract N* points from a uniform distribution with support on the entire range of possible values, that is estimated by fitting N distinct linear regressions for each one of the N groups, as before, and identifying the minimum and the maximum values;
- we uniformly distribute the weights on these N* support points.

During the iterations, the EM algorithm performs the support reduction of the discrete distribution, in order to identify $M < N$ mass points in which the N groups are clustered. The support reduction is made standing on two criteria. The former is that we fix a threshold D and if two points $\mathbf{c}_l$ and $\mathbf{c}_k$ are closer than D, in terms of euclidean distance, they collapse to a unique point $\mathbf{c}_{l,k}$, where $\mathbf{c}_{l,k} = \frac{\mathbf{c}_l + \mathbf{c}_k}{2}$ with weight $w_{l,k} = w_l + w_k$. The first two masses collapsing to a unique point are the two masses with the minimum euclidean distance, among the couples of masses with euclidean distance less than D, and so on so forth. The latter is that, starting from a given iteration up to the end, we fix a threshold $\tilde{w}$ and we remove mass points with weight $w_l \leq \tilde{w}$ or that are not associated to any subpopulation. D and $\tilde{w}$ are two tuning parameters that tune the estimates of the subpopulations. The choice of D depends on how much we want to be sensitive to the differences among subpopulations: the higher is D, the lower is the number of subpopulations and the less homogeneous are the groups within subpopulations. D depends also on the order of magnitude of the data. The choice of $\tilde{w}$ depends on the minimum number of groups that we allow within each subpopulation. When one or more mass points are deleted, the remaining weights are reparameterized in such a way that they sum up to 1:

$$\begin{aligned} S_w &= \sum_{l=1}^{M^{new}} w_l^{old} \\ w_l^{new} &= \frac{w_l^{old}}{S_w} \quad \forall l = 1, \ldots, M^{new} \end{aligned} \tag{2.19}$$

where M^{new} is the total number of masses after deleting the ones associated to weight $w_l \leq \tilde{w}$ or not associated to any subpopulation, $\mathbf{w}^{old}$ are the old remaining weights and $\mathbf{w}^{new}$ are the new reparameterized weights.

Algorithm 2.2 EM algorithm for semi-parametric mixed-effects models

input : Initial estimates for $(\mathbf{c}_1^{(0)}, \ldots, \mathbf{c}_M^{(0)})$ and $(w_1^{(0)}, \ldots, w_M^{(0)})$, with $M = N$;
Initial estimates for $\beta^{(0)}$ and $\sigma^{2(0)}$;
Tolerance parameters D, $\tilde{w}$, tollR, tollF, it, it1, itmax.

output: Final estimates of $\mathbf{c}_l^{(it)}$, $w_l^{(it)}$, for $l = 1, \ldots, M$, $\beta^{(it)}$ and $\sigma^{2(it)}$.

k=1; conv1=0; conv2=0;

while *(conv*1 == 0 *or conv*2 == 0 & $k < it$*)* **do**

compute the distance matrix DIST (where $DIST_{st} = \sqrt{(c_{0s} - c_{0t})^2 + (c_{1s} - c_{1t})^2}$ is the euclidean distance between each couple of mass points $s, t \; \forall s, t = 1, \ldots, M, s \neq t$);

if *(*$DIST_{st} < D$ & $DIST_{st} = min(DIST)$ $\quad (\forall s, t = 1, \ldots, M, s \neq t)$*)* **then**

collapse masses s and t to a unique mass point;

compute the new distance matrix DIST;

if *conv*1 == 1 *or* $k \geq it1$ **then**

if $w_l^{(k)} \leq \tilde{w} \quad (\forall l = 1, \ldots, M)$ **then**

delete mass point l;

reparameterize the weights according to Eq. (2.19);

if *no changes are done* **then**

conv2=1;

given $\mathbf{c}_l^{(k-1)}, w_l^{(k-1)}$ for $l = 1, \ldots, M$, $\beta^{(k-1)}$ and $\sigma^{2(k-1)}$, compute the matrix W according to Eq. (2.10);

update the weights $w_1^{(k)}, \ldots, w_M^{(k)}$ according to Eq. (2.8);

$\beta^{(k,0)} = \beta^{(k-1)}$;
$\sigma^{2(k,0)} = \sigma^{2(k-1)}$;
$\mathbf{c}_l^{(k,0)} = \mathbf{c}_l^{(k-1)}$;
$w_l^{(k,0)} = w_l^{(k-1)}$;

keeping $\beta^{(k,0)}$ and $\sigma^{2(k,0)}$ fixed, update the M support points $\mathbf{c}_1^{(k,1)}, \ldots, \mathbf{c}_M^{(k,1)}$ according to Eq. (2.13) and (2.14);

keeping $\mathbf{c}_l^{(k,1)}, w_l^{(k,0)}$ for $l = 1, \ldots, M$ fixed, update $\beta^{(k,1)}$ and $\sigma^{2(k,1)}$ according to Eq. (2.15) and (2.16);

j=1;

while $(|\beta^{(k,j-1)} - \beta^{(k,j)}| \geq tollF \quad or \quad |\sigma^{2(k,j-1)} - \sigma^{2(k,j)}| \geq tollF \quad or \quad |\mathbf{c}_l^{(k,j-1)} - \mathbf{c}_l^{(k,j)}| \geq tollR) \quad \& \quad j \leq itmax$ **do**

j=j+1; keeping $\beta^{(k,j-1)}$ and $\sigma^{2(k,j-1)}$ fixed, update the M support points $\mathbf{c}_1^{(k,j)}, \ldots, \mathbf{c}_M^{(k,j)}$ according to Eq. (2.13) and (2.14);

keeping $\mathbf{c}_l^{(k,j)}, w_l^{(k,j-1)}$ for $l = 1, \ldots, M$ fixed, update $\beta^{(k,j)}$ and $\sigma^{2(k,j)}$ according to Eq. (2.15) and (2.16);

set $\mathbf{c}_l^{(k)} = \mathbf{c}_l^{(k,j)}$ for $l = 1, \ldots, M$, $\beta^{(k)} = \beta^{(k,j)}$, $\sigma^{2(k)} = \sigma^{2(k,j)}$;

estimate subpopulation l for each group i according to Eq. (2.18);

if $(\beta^{(k)} - \beta^{(k-1)} < tollF)$ & $(\sigma^{2(k)} - \sigma^{2(k-1)} < tollF)$ & $(\mathbf{c}_l^{(k)} - \mathbf{c}_l^{(k-1)} < tollR)$ **then**

conv1=1;

k= k+1;

2.2.2. Semi-parametric mixed-effects models for a bivariate continuous response

In this subsection we show the extension of the SPEM method to the case of a bivariate response variable.

Consider a bivariate two-level linear model, where each bivariate observation j, for $j = 1, \ldots, n_i$, is nested within a group i, for $i = 1, \ldots, N$. The model takes the following form:

$$\begin{pmatrix} \mathbf{y}_{1,i} \\ \mathbf{y}_{2,i} \end{pmatrix}^T = \mathbf{X}_i \begin{pmatrix} \boldsymbol{\beta}_1 \\ \boldsymbol{\beta}_2 \end{pmatrix}^T + \mathbf{Z}_i \begin{pmatrix} \boldsymbol{\delta}_{1,i} \\ \boldsymbol{\delta}_{2,i} \end{pmatrix}^T + \begin{pmatrix} \boldsymbol{\epsilon}_{1,i} \\ \boldsymbol{\epsilon}_{2,i} \end{pmatrix}^T \quad i = 1, \ldots, N, \tag{2.20}$$

$$\boldsymbol{\epsilon}_i^T = \begin{pmatrix} \boldsymbol{\epsilon}_{1,i} \\ \boldsymbol{\epsilon}_{2,i} \end{pmatrix} \sim \mathcal{N}_2(\mathbf{0}, \boldsymbol{\Sigma}) \quad \text{ind .}$$

where :

$-\mathbf{Y}_i = \begin{pmatrix} y_{1,1i}, \ldots, y_{1,n_i i} \\ y_{2,1i}, \ldots, y_{2,n_i i} \end{pmatrix}^T$ is the $(n_i \times 2)$-dimensional matrix of response variable within the i-th second level group [3],

$-\mathbf{X}_i$ is the $(n_i \times (P+1))$-dimensional matrix of covariates relative to fixed coefficients,

$-\mathbf{B} = \begin{pmatrix} \boldsymbol{\beta}_1 & \boldsymbol{\beta}_2 \end{pmatrix}$ is the $((P+1) \times 2)$-dimensional matrix of coefficients of $\mathbf{X}$,

$-\mathbf{Z}_i$ is the $(n_i \times (R+1))$-dimensional matrix of covariates relative to random coefficients,

$-\mathbf{1}_i = \begin{pmatrix} \boldsymbol{\delta}_{1,i} & \boldsymbol{\delta}_{2,i} \end{pmatrix}$ is the $((R+1) \times 2)$-dimensional matrix of random coefficients of $\mathbf{Z}_i$

$-\boldsymbol{\epsilon}_i = \begin{pmatrix} \boldsymbol{\epsilon}_{1,i} & \boldsymbol{\epsilon}_{2,i} \end{pmatrix}$ is the $(n_i \times 2)$-dimensional matrix of errors and $\boldsymbol{\Sigma}$ is its variance/covariance matrix.

For each response variable, this parametric distribution allows to associate each group i to a different set of coefficients $\boldsymbol{\delta}_{*,i} = \left(\delta_{*,i1}, \ldots, \delta_{*,i(R+1)}\right)$, where * can be eqaul to 1 or 2, for the $(R+1)$ covariates of the random effects, extracted from the normal distribution.

In subscript of each variable/parameter, we indicate by the number before the comma whether the variable/parameter is referred to the first or the second response variable (for example, $y_{1,ji}$ and $y_{2,ji}$ are the j-th first and second response variables within (level 2)-group i, respectively).

As with the previous case we relax the parametric assumptions about the coefficients of

the random effects and we assume the bivariate coefficients $\mathbf{1}_i = \begin{pmatrix} \boldsymbol{\delta}_{1,i} & \boldsymbol{\delta}_{2,i} \end{pmatrix}$ to follow a bivariate discrete distribution S^*, assuming $M \times K$ mass points $(\mathbf{C}_{11}, \ldots, \mathbf{C}_{MK})$, where each $\mathbf{C}_{mk}$ is the $2 \times (R+1)$-dimensional matrix of coefficients of random effects for the bivariate mass point related to the index (m, k), for each $m = 1, \ldots, M$ and $k = 1, \ldots, K$, where both M and K are smaller than N. The total number of mass points, that is $M \times K$, is unknown a priori and it is estimated together with the other parameters of the model. This modelling allows the identification of a bivariate clustering distribution among the N groups, where each group i is associated to a bivariate cluster, standing on the linear relationships between the two response variables and their covariates. In other words, the model identifies a bivariate latent structure among the groups, that also reveals the dependence among the two response variables. Under these assumptions, the semi-parametric bivariate model with random coefficients takes the following form:

$$\begin{gathered} \begin{pmatrix} \mathbf{y}_{1,i} \\ \mathbf{y}_{2,i} \end{pmatrix}^T = \mathbf{X}_i \begin{pmatrix} \boldsymbol{\beta}_1 \\ \boldsymbol{\beta}_2 \end{pmatrix}^T + \mathbf{Z}_i \begin{pmatrix} \mathbf{c}_{1,m} \\ \mathbf{c}_{2,k} \end{pmatrix}^T + \begin{pmatrix} \boldsymbol{\epsilon}_{1,i} \\ \boldsymbol{\epsilon}_{2,i} \end{pmatrix}^T \\ i = 1, \ldots, N \quad m = 1, \ldots, M \quad k = 1, \ldots, K \\ \boldsymbol{\epsilon}_i^T = \begin{pmatrix} \boldsymbol{\epsilon}_{1,i} \\ \boldsymbol{\epsilon}_{2,i} \end{pmatrix} \sim \mathcal{N}_2(\mathbf{0}, \boldsymbol{\Sigma}) \quad \text{ind} \, . \end{gathered} \tag{2.21}$$

Even in this case the algorithm follows the idea presented in [17] considering, without loss of generality, the case of a semi-parametric bivariate twolevel linear model, with one random intercept, one random covariate and P fixed covariates [4], reducing the previous model in the following form:

$$\begin{gathered} \begin{pmatrix} \mathbf{y}_{1,i} \\ \mathbf{y}_{2,i} \end{pmatrix}^T = \mathbf{1}_{n_i} \begin{pmatrix} c_{1,1m} \\ c_{2,1k} \end{pmatrix}^T + \sum_{p=1}^{P} \mathbf{x}_{ip} \begin{pmatrix} \boldsymbol{\beta}_{1p} \\ \boldsymbol{\beta}_{2p} \end{pmatrix}^T + \mathbf{z}_i \begin{pmatrix} c_{1,2m} \\ c_{2,2k} \end{pmatrix}^T + \begin{pmatrix} \boldsymbol{\epsilon}_{1,i} \\ \boldsymbol{\epsilon}_{2,i} \end{pmatrix}^T \\ i = 1, \ldots, N \quad m = 1, \ldots, M \quad k = 1, \ldots, K \\ \boldsymbol{\epsilon}_i^T = \begin{pmatrix} \boldsymbol{\epsilon}_{1,i} \\ \boldsymbol{\epsilon}_{2,i} \end{pmatrix} \sim \mathcal{N}_2(\mathbf{0}, \boldsymbol{\Sigma}) \quad \text{ind} \end{gathered} \tag{2.22}$$

where $\mathbf{1}_{n_i}$ is the n_i-dimensional vector of 1, M is the total number of mass points for the first response and K is the total number of mass points for the second response and both of them are unknown a priori. Coefficients $\mathbf{C}_{mk}$, for $m = 1, \ldots, M$ and $k = 1, \ldots, K$ are distributed according to a discrete probability measure S^* that belongs to the class of all

probability measures on $\mathcal{R}^4$. S^* can then be interpreted as the mixing distribution that generates the density of the stochastic model in 2.22. For further details it is possible to look at the Algorithm 2.3

Algorithm 2.3 EM algorithm for bivariate semi-parametric models with random coefficients

input : Initial estimates for $(\mathbf{C}_{11}^{(0)}, \dots, \mathbf{C}_{MK}^{(0)})$ and $(w_{11}^{(0)}, \dots, w_{MK}^{(0)})$, with $M = N$ and $K = N$;
Initial estimates for $\boldsymbol{B}^{(0)}$ and $\boldsymbol{\Sigma}^{(0)}$;
Tolerance parameters D_1, D_2, $\tilde{w}_1$, $\tilde{w}_2$, `tollR`, `tollF`, `it`, `it1`, `itmax`.
output: Final estimates of $\mathbf{C}_{mk}^{(a)}$, $w_{mk}^{(a)}$, for $m = 1, \dots, M$, $k = 1, \dots, K$, $\boldsymbol{B}^{(a)}$ and $\boldsymbol{\Sigma}^{(a)}$.
a=1; conv1=0; conv2=0;

while *(conv*1 == 0 *or conv*2 == 0 & $a <$ `it`*)* **do**
  compute the distance matrices DIST1 and DIST2 for both the subpopulations distribution (where, e.g.,for the first response variable, $DIST1_{st} = \sqrt{(c_{1,1s} - c_{1,1t})^2 + (c_{1,2s} - c_{1,2t})^2}$ is the euclidean distance between each couple of mass points s, t $\forall s, t = 1, \dots, M, s \neq t$);
  if *(*$DIST1_{st} < D_1$ & $DIST1_{st} = min(DIST1)$ $\quad (\forall s, t = 1, \dots, M, s \neq t)$*)* **then**
    collapse marginal masses s and t to a unique mass point;
  if *(*$DIST2_{st} < D_2$ & $DIST2_{st} = min(DIST2)$ $\quad (\forall s, t = 1, \dots, K, s \neq t)$*)* **then**
    collapse marginal masses s and t to a unique mass point;
  compute the new distance matrices DIST1 and DIST2;

  if *conv*1 == 1 *or* $a \geq$ `it1` **then**
    if $w_{1,m}^{(a)} \leq \tilde{w}_1 \quad (\forall m = 1, \dots, M)$ **then**
      delete marginal mass point m;
      reparameterize the weights;
    if $w_{2,k}^{(a)} \leq \tilde{w}_2 \quad (\forall k = 1, \dots, K)$ **then**
      delete marginal mass point k;
      reparameterize the weights;
    if *no changes are done* **then**
      *conv*2 = 1;

  given $\mathbf{C}_{mk}^{(a-1)}, w_{mk}^{(a-1)}$ for $m = 1, \dots, M$ and $k = 1, \dots, K$, $\boldsymbol{B}^{(a-1)}$ and $\boldsymbol{\Sigma}^{(a-1)}$, compute the matrix W according to Eq. (**??**);
  update the weights $w_{11}^{(a)}, \dots, w_{MK}^{(a)}$ according to Eq. (2.8);

  $\boldsymbol{B}^{(a,0)} = \boldsymbol{B}^{(a-1)}$;
  $\boldsymbol{\Sigma}^{(a,0)} = \boldsymbol{\Sigma}^{(a-1)}$;
  $\mathbf{C}_{mk}^{(a,0)} = \mathbf{C}_{mk}^{(a-1)}$;
  $w_{mk}^{(a,0)} = w_{mk}^{(a-1)}$;

  keeping $\boldsymbol{B}^{(a,0)}$ and $\boldsymbol{\Sigma}^{(k,0)}$ fixed, update the $M \times K$ support points $\mathbf{C}_{11}^{(a,1)}, \dots, \mathbf{C}_{MK}^{(a,1)}$ according to Eq. (2.9);
  keeping $\mathbf{C}_{mk}^{(a,1)}, w_{mk}^{(a,0)}$ for $m = 1, \dots, M$ and $k = 1, \dots, K$ fixed, update $\boldsymbol{B}^{(a,1)}$ and $\boldsymbol{\Sigma}^{(a,1)}$ according to Eq. (2.9);

  j=1;
  while $(|\boldsymbol{B}^{(a,j-1)} - \boldsymbol{B}^{(a,j)}| \geq tollF \quad or \quad |\boldsymbol{\Sigma}^{(a,j-1)} - \boldsymbol{\Sigma}^{(a,j)}| \geq tollF \quad or \quad |\mathbf{C}_{mk}^{(a,j-1)} - \mathbf{C}_{mk}^{(a,j)}| \geq tollR) \quad \& \quad j \leq$ `itmax` **do**
    j=j+1; keeping $\boldsymbol{B}^{(a,j-1)}$ and $\boldsymbol{\Sigma}^{(a,j-1)}$ fixed, update the $M \times K$ support points $\mathbf{C}_{11}^{(a,j)}, \dots, \mathbf{C}_{MK}^{(a,j)}$ according to Eq. (2.9);
    keeping $\mathbf{C}_{mk}^{(a,j)}, w_{mk}^{(a,j-1)}$ for $m = 1, \dots, M$ and $k = 1, \dots, K$ fixed, update $\boldsymbol{B}^{(a,j)}$ and $\boldsymbol{\Sigma}^{(a,j)}$ according to Eq. (2.9);

  set $\mathbf{C}_{mk}^{(a)} = \mathbf{C}_{mk}^{(a,j)}$ for $m = 1, \dots, M$ and $k = 1, \dots, K$, $\boldsymbol{B}^{(a)} = \boldsymbol{B}^{(a,j)}$, $\boldsymbol{\Sigma}^{(a)} = \boldsymbol{\Sigma}^{(a,j)}$;
  estimate subpopulation mk for each group i according to Eq. (2.18);
  if $(\boldsymbol{B}^{(a)} - \boldsymbol{B}^{(a-1)} < tollF)$ & $(\boldsymbol{\Sigma}^{(k)} - \boldsymbol{\Sigma}^{(k-1)} < tollF)$ & $(\mathbf{C}_{mk}^{(a)} - \mathbf{C}_{mk}^{(a-1)} < tollR)$ **then**
    *conv*1 = 1;
  a= a+1;

2.2.3. Semi-parametric mixed-effects models for a multinomial response

In this section we describe the case int which the response variable has a multinomial form. Let's consider a multinomial logistic regression model for nested data with a two-level hierarchy [1**?**], where each observation j, for $j = 1, \dots, n_i$, is nested within a group i, for $i = 1, \dots, I$. Let $\mathbf{Y}_i = (Y_{i1}, \dots, Y_{in_i})$ be the n_i-dimensional response vector for observations within the $i-$th group. The multinomial distribution with K categories relative to Y_{ij} is the following:

$$Y_{ij} = \begin{cases} 1 & \pi_{ij1} \\ 2 & \pi_{ij2} \\ \dots & \\ K & \pi_{ijK} \end{cases}, \tag{2.23}$$

where $k = 1, \dots, K$ indexes the K support points of the discrete distribution of Y_{ij} and π_{ijk} is the probability that observation j within group i assumes value k. Mixed-effects multinomial models assume that the probability that $Y_{ij} = k$, i.e. π_{ijk}, is given by

$$\begin{cases} \pi_{ijk} = P(Y_{ij} = k) = \frac{\exp(\eta_{ijk})}{1+\sum_{k=2}^{K}\exp(\eta_{ijk})} & \text{for} \quad k = 2, \dots, K \\ \pi_{ij1} = P(Y_{ij} = 1) = \frac{1}{1+\sum_{k=2}^{K}\exp(\eta_{ijk})} \end{cases}, \tag{2.24}$$

where $\eta_{ijk} = \mathbf{x}'_{ij}\boldsymbol{\alpha}_k + \mathbf{z}'_{ij}\boldsymbol{\delta}_{ik}$ is the linear predictor. $\mathbf{x}_{ij}$ is the $p \times 1$ covariates vector (includes a 1 for the intercept) of the fixed effects, $\boldsymbol{\alpha}_k$ is the $p \times 1$ vector of regression parameters of the fixed effects, $\mathbf{z}_{ij}$ is the $q \times 1$ covariates vector of the random effects (includes a 1 for the intercept) and $\boldsymbol{\delta}_{ik}$ is the $q \times 1$ vector of regression parameters of the random effects. Logit models for nominal response basically pair each category with a baseline category. This formulation considers $K - 1$ contrasts, between each category k, for $k = 2, \dots, K$, and the reference category[1], that is $k = 1$. Consequently, each category is assumed to be related to a latent "response tendency" for that category with respect to the reference one. Each contrast $k', k' = 1, \dots, K - 1$, is characterized by the set of *contrast-specific* parameters $(\boldsymbol{\alpha}_{k'}; \boldsymbol{\delta}_{ik'}$, for $i = 1, \dots, I)$, that models the probability of Y_{ij} being equal to $k \equiv k' + 1$ with respect to the probability of Y_{ij} being equal to 1 (the

[1]We consider the first category as the reference one but this choice is arbitrary and it does not affect the model formulation.

reference category)[2]. Starting from Eq. (2.24), the log-odds of each response with respect to the reference category are:

$$\log\left(\frac{\pi_{ijk}}{\pi_{ij1}}\right) = \eta_{ijk} \qquad k = 2, \ldots, K. \tag{2.25}$$

For each contrast, the *contrast-specific* random-effects parameters describe the latent structure at the highest level of the hierarchy.

The Maximum Likelihood Estimation (MLE) method allows to estimate the model parameters of this probability distribution.

Considering $\mathbf{A} = (\boldsymbol{\alpha}_2, \ldots, \boldsymbol{\alpha}_K)$ and $\boldsymbol{\Delta}_i = (\boldsymbol{\delta}_{i2}, \ldots, \boldsymbol{\delta}_{iK})$, the distribution of Y_{ij}, conditional on the random effects distribution, takes the following form:

$$\begin{aligned} p(Y_{ij}|\mathbf{A}, \boldsymbol{\Delta}_i) =& \pi_{ij1}^{\mathbf{1}_{\{Y_{ij}=1\}}} \times \pi_{ij2}^{\mathbf{1}_{\{Y_{ij}=2\}}} \times \ldots \times \pi_{ijK}^{\mathbf{1}_{\{Y_{ij}=K\}}} = \\ =& \prod_{k=1}^{K} \pi_{ijk}^{\mathbf{1}_{\{Y_{ij}=k\}}} = \\ =& \prod_{k=1}^{K} \left(\frac{\exp(\eta_{ijk})}{1 + \sum_{l=2}^{K} \exp(\eta_{ijl})}\right)^{\mathbf{1}_{\{Y_{ij}=k\}}}. \end{aligned} \tag{2.26}$$

Assuming that Y_{ij} and $Y_{ij'}$ are independent for $j \neq j'$, the conditional distribution of $\mathbf{Y}_i$ is:

$$\begin{aligned} p(\mathbf{Y}_i|\mathbf{A}, \boldsymbol{\Delta}_i) =& \frac{\left(\sum_{k=1}^{K}\left(\sum_{j=1}^{n_i} \mathbf{1}_{\{Y_{ij}=k\}}\right)\right)!}{\prod_{k=1}^{K}\left(\left(\sum_{j=1}^{n_i} \mathbf{1}_{\{Y_{ij}=k\}}\right)!\right)} \times \prod_{j=1}^{n_i} p(Y_{ij}|\mathbf{A}, \boldsymbol{\Delta}_i) = \\ =& \frac{\left(\sum_{k=1}^{K}\left(\sum_{j=1}^{n_i} \mathbf{1}_{\{Y_{ij}=k\}}\right)\right)!}{\prod_{k=1}^{K}\left(\left(\sum_{j=1}^{n_i} \mathbf{1}_{\{Y_{ij}=k\}}\right)!\right)} \times \prod_{j=1}^{n_i}\prod_{k=1}^{K} \pi_{ijk}^{\mathbf{1}_{\{Y_{ij}=k\}}} = \\ =& \frac{\left(\sum_{k=1}^{K}\left(\sum_{j=1}^{n_i} \mathbf{1}_{\{Y_{ij}=k\}}\right)\right)!}{\prod_{k=1}^{K}\left(\left(\sum_{j=1}^{n_i} \mathbf{1}_{\{Y_{ij}=k\}}\right)!\right)} \times \prod_{j=1}^{n_i}\prod_{k=1}^{K} \left(\frac{\exp(\eta_{ijk})}{1 + \sum_{l=2}^{K} \exp(\eta_{ijl})}\right)^{\mathbf{1}_{\{Y_{ij}=k\}}}. \end{aligned} \tag{2.27}$$

Even in this case we move to a semi parametric framework by assuming the coefficients of the random effects to follow a discrete distribution with an *a priori* unknown number

[2]Note that $k' \equiv k - 1$ for $k = 2, \ldots, K$ and, therefore the sequence of parameters $(\boldsymbol{\alpha}_{k'}; \boldsymbol{\delta}_{ik'}$, for $i = 1, \ldots, I)$, for $k' = 1, \ldots, K-1$ is equal to the sequence $(\boldsymbol{\alpha}_k; \boldsymbol{\delta}_{ik}$, for $i = 1, \ldots, I)$ for $k = 2, \ldots, K$.

of support points. Under this assumption, the multinomial logit takes the form:

$$\eta_{ijk} = \mathbf{x}'_{ij}\boldsymbol{\alpha}_k + \mathbf{z}'_{ij}\mathbf{b}_{m_k k} \qquad m_k = 1, \ldots, M_k, \quad k = 2, \ldots, K, \tag{2.28}$$

where M_k is the total number of support points of the discrete distribution of $\mathbf{b}$ relative to the $k-$th category, for $k = 2, \ldots, K$. The random effects distribution relative to each category k, for $k = 2, \ldots, K$, can be expressed as a set of points $(\mathbf{b}_{1k}, \ldots, \mathbf{b}_{M_k k})$, where $M_k \leq I$ and $\mathbf{b}_{m_k k} \in \mathcal{R}^q$ for $m_k = 1, \ldots, M_k$, and a set of weights $(w_{1k}, \ldots, w_{M_k k})$, where $\sum_{m_k=1}^{M_k} w_{m_k k} = 1$ and $w_{m_k} \geq 0$:

$$\mathbf{B} = \begin{cases} \begin{cases} \mathbf{b}_{12}, \quad \mathbf{b}_{22}, \ldots, \quad \mathbf{b}_{M_2 2} \\ (w_{12}), (w_{22}), \ldots, (w_{M_2 2}) \end{cases} \\ \ldots \\ \ldots \\ \begin{cases} \mathbf{b}_{1K}, \quad \mathbf{b}_{2K}, \ldots, \quad \ldots, \quad \mathbf{b}_{M_K K} \\ (w_{1K}), (w_{2K}), \ldots, \quad \ldots, (w_{M_K K}) \end{cases} \end{cases} . \tag{2.29}$$

The discrete distributions P_k^*, for $k = 2, \ldots, K$, belong to the class of all probability measures on $\mathcal{R}^q$ and are assumed to be independent. P_k^* is a discrete measure with M_k support points that can then be interpreted as the mixing distribution that generates the density of the stochastic model in Eq.(2.28). In particular, $w_{m_k k} = P(\boldsymbol{\delta}_{ik} = \mathbf{b}_{m_k k})$, for $i = 1, \ldots, I$.

Given this formulation two patths open up, in [18] the authors propose the *MSPEM* algorithm for the joint estimations of $\boldsymbol{\alpha}_k$, $(\mathbf{b}_{1k}, \ldots, \mathbf{b}_{M_k k})$ and $(w_{1k}, \ldots, w_{M_k k})$, for $k = 2, \ldots, K$, which is performed through the maximization of the likelihood, mixture by the discrete distribution of the random effects. In the *MSPEM* steps, under the independence assumption across the contrast-specific random-effects, when estimating the support points relative to each contrast, the other contrast-specific random-effects parameters are fixed to the mean of the relative discrete distributions. In other words, when estimating the random effects of a group with respect to a response category, the random effects of this specific group with respect to the other categories are ignored. This assumption simplifies the parameters estimation procedure, but it is often too strict and unrealistic. For this reason the authors in [19] propose a different method in which the independence across the random effects distributions relative to the $(K-1)$ categories is not assumed. The JMSPEM method allows for greater flexibility therefore it is preferred to MSPEM in

this thesis. The functions described in later chapters will refer to the method with joint estimates. More details regarding the algorithms, differences between the two methods, and further theoretical details can be found in the articles [18] and [19]

3 | Package structure and usage

In this section all the instructions useful to the user to use the package are presented. To explain both the command useful to run the models and the default values of the variables, a definition of the `R` function for each statistical model is provided. Next, the output of the function is shown, followed by a list of all auxiliary functions. The explanation of these functions will be differentiated for those related to the first branch of research while for the second branch they will be explained only once given the small amount of differences. It is significant to note that no relevant examples are provided for either the auxiliary functions or outputs. This topic is postponed to the following parts because the next chapter is concentrated on providing examples relevant to all the statistical models discussed. The analysis of some parameters related to the convergence of iterative algorithms is also referred to the next chapter. In fact, some significant variations related to the change of parameters will be presented so only the theoretical definition of the parameter is presented below. The installation of the package is made through the following line of command

```
install.packages("SpMEMs")
```

. and then loaded to the workspace thorugh th following command:

```
library("SpMEMs")
```

3.1. Generalised mixed effect Tree

The R function relative to the GMET function is the following:

```
GMET(formula, dataset, random, subset = NULL,
                          family = binomial,
                          tree.control =rpart.control(),
                          cv=TRUE, cpmin = 0.01, verbose = FALSE)
```

The three necessary inputs are `formula dataset` and `random` while the other ones are already initialized and fine-tune the algorithm. The description of the inputs is the fol-

lowing

- `formula`: a formula expression as for regression models, of the form `response` $\sim$ `predictors`, for example: $y \sim x + z$. The predicotrs variable must contain only the name of the covariate that has to be treated as fixed.
- `dataset`: a data frame, list or environment in which to interpret the variables occurring in `formula`.
- `random`: a formula expression for the random effects. Those terms are distinguished by vertical bars ('|') separating expressions for design matrices from grouping factors.
- `subset`: a vector of index indicating the rows of the dataset that has to be used to train the random forest. If the input is not filled all the dataset is used as training dataset.
- `family`: a GLM family of the response, consult [28] for further details.
- `tree.control` :a list of options that control details of the Recursive Partitioning and Regression Trees (rpart) algorithm. All the informations are contained in the documentation of the `rpart` package [33].
- `cv`: a logical parameter. If it is equal to TRUE the cross validation is performed.
- `cpmin`: complexity parameter. Any split that does not decrease the overall lack of fit by a factor of `cp` is not attempted. For instance, with anova splitting, this means that the overall R-squared must increase by `cp` at each step. The main role of this parameter is to save computing time by pruning off splits that are obviously not worthwhile. Essentially, the user informs the program that any split which does not improve the fit by `cp` will likely be pruned off by cross-validation, and that hence the program need not pursue it.[33]
- `verbose`: a logical parameter. If it is equal to TRUE the summary of the general linear model used to initialize the fixed effect is displayed.

The output of the function is an object of the `my.mixed.tree` class that is composed as follow:

- `Tree`: a `rpart` object that describes the tree object used to estimate the fixed effects [33].
- `EffectModel`: a `merMod` object containing the result of the Generalized linear mixed-effect model obtained witht the tree object estimated in the function as fixed effect. For further information about the functioning of a `merMod` consult the [2]

- `RandomEffects`: a `ranef.mer` object containing a dataframe composed by the random effect parameters for each group estimated in the model.
- `BetweenMatrix`: an object of class `VarCorr.merMod` containing the estimated variances, standard deviations, and correlations between the random-effects terms in a mixed-effects model.

Below the auxiliary functions of the statiscal model are presented.

3.1.1. Recursive Partitioning and Regression Trees plot

The first auxiliary function is a graphical function. It allows to visualize the `rpart` plot relative to the estimated mixed-effects tree. In this case the tree-object is used to estimate the fixed effect part of the model i.e. the $\boldsymbol{f}(X_i)$ function of the model in Eq.(2.2). The generic call of the function is the following:

```
plot.mymixedtree(x)
```

3.1.2. Dotplot of the estimated random intercept

The second function is also graphical therefore it consists of a graphical tool aimed at displaying no longer the fixed component but rather the random component of the model presented in Eq.(2.2). It allows the visualization of the estimated random intercept toghether with the 95% confidence interval for each group of the random effect. The generic call of the function is the following:

```
dotplot.GMET(gmt,rand_int=FALSE)
```

`gmt` is a GMET object while `rand_int` is a logical value. If it is equal to `TRUE` the function displays the random intercept of the model.

3.1.3. GMET prediction function

The third function we present is no longer graphical but it gives the possibility to know the predicted value of a new observation through the model obtained using the GMET algorithm. The predict function has the following generic call:

```
predict.mymixedtree(object, newdata, type = "response")
```

where:

- `object`: is a GMET object.

- `newdata`:is a data frame where one can search for variables to use in predictions. The dataframe's columns must bear the same names as those of the `dataset` used in the call of `GMET` function.
- `type`: the type of prediction required. The default is on the scale of the linear predictors; the alternative "response" is on the scale of the response variable. Thus for a default binomial model the default predictions are of log-odds (probabilities on logit scale) and type = "response" gives the predicted probabilities. The "terms" option returns a matrix giving the fitted values of each term in the model formula on the linear predictor scale.

3.2. Generalised mixed effect Random Forest

Let us consider a different version of the previous algorithm in which we use a random forest model as a tree-object rather than a non parametric CART model for estimating the $\boldsymbol{f}(X_i)$ function of the model presented in Eq.(2.2) The GMERF function can be invoked with the following code

```
GMERF(y, cov, group, xnam=NULL, znam=NULL,
        family='binomial', bizero=NULL,
        itmax=30, toll=0.02)
```

The explanation of the various input is the following:

- `y`: is a vector containing the response variable of the model.
- `cov`: a dataframe object containing all the covariates that has to be considered in the model. It is important to note that it is not necessary to add a column of ones in the `cov` dataframe since the code does it automatically.
- `group`: a factor element that contains the information about the group of the observations contained in `y` and `cov`.
- `xnam`: an array with the names of the columns of the `cov` dataset that has to be considered as fixed elements.
- `znam`: an array with the names of the columns of the `cov` dataset that has to be considered as random elements.
- `family`: a GLM family, consult [28].
- `bizero`: a matrix of dimension $(q+1) \times N$ for the initialization of the random

effects parameters. The first column is the intercept parameter while the other are the ones concerning the other `znam` covariates. The default option creates a matrix with a vector of ones in the first row and zeros in the other elements.

- `itmax`: a integer that indicates the maximum number of the iteration performed by the algorithm.
- `toll`: a real number that indicates the minimum difference between $\boldsymbol{b}_i$'s of two consecutive iterations.

It is useful to highlight the fact that the `group` variable assume consistency with the `bizero` matrix. The `i` column of the matrix `bizero` corresponds to the `i` group level of the variable `group`. The call of the GMERF function generates a GMERF object that has the following composition:

- `glmer.model`: a `merMod` object containing the result of the Generalized linear mixed-effect model obtained with the tree object estimated in the function as fixed effect. For further information about the functioning of a `merMod` consult the [2].
- `forest`: an object of class `randomForest` that describes the result of the estimation of the fixed effect function used in the model. For further information about the `randomForest` it is possible to consult the package documentation of the `R` package `rpart` [33].
- `rand.coef`: a dataframe of dimension $(q + 1)$ containing the estimated random coefficient of the model , i.e. the $\mathbf{b}_i$ of the model presented in Eq.(2.2).
- `n.iteration`: an integer that indicates how many iterations the algorithm did.
- `converged`: a logical value that indicates if the algorithm reached convergence or if it stopped at the maximum number iteration indicated as input.
- `all.rand.coef`: a list of dataframes containing the $\mathbf{b}_i$'s obtained at every iteration.
- `linkf`: a function object containing the link function used in the model.
- `linkinv`: a function object containing the inverse of the link function used in the model.
- `forest.var`: a vector of string containing the names of the covariates used to obtain the random forest object.
- `random.eff.var`: a string vector providing the name of the covariates that was utilized to create the random forest object.

- `family`: a string containing the type of `family` used in the model.

There are 5 auxiliary functions useful to the user in understanding the output of the algorithm.

3.2.1. Variable Importance Plot

The second auxiliary is a graphical function too. When dealing with a general random forest, it is possibile to understand the importance of a variable in the prediction of the response using the Variable Importance Plot (VIP). For further information about the VIP it is possible to consult the R documentation of the `VarImpPlot` of the `randomForest` package in [11] The generic call of the function is thee following:

```
VIP.GMERF(gmf,covnames=NULL)
```

where `gmf` is a GMERF object while `covnames` is a string containing the names of the covariates of the model. If `covnames` is not filled the names of the covariates will not be displayed in the x-axes of the plot.

3.2.2. Dotplot of the estimated random intercept

The first function is a graphical tool aimed at displaying the random component of the model presented in Eq.(2.2). It allows the visualization of the estimated random intercept toghether with the 95% confidence interval for each group of the random effect. The generic call of the function is the following:

```
dotplot.GMERF(gm,rand_int=FALSE)
```

where `gmf` is a GMERF object while `rand_int` is a logical value. If it is equal to `TRUE` the function displays the random intercept of the model.

3.2.3. GMERF fitted function

The third auxiliary function allow to obtain the fitted values of the algorithm. That are the predicted values of the response obtained using as independent covariates the dataset given as input in the GMERF function. The generic call of the function is:

```
fitted.GMERF(gmf, type='response', alpha=0.5)
```

where :

- `gm`:is a GMERF object

- `type`:is the type of prediction required. The default is on the scale of the linear predictors; the alternative "response" is on the scale of the response variable. Thus for a default binomial model the default predictions are of log-odds (probabilities on logit scale) and type = "response" gives the predicted probabilities. The "terms" option returns a matrix giving the fitted values of each term in the model formula on the linear predictor scale.
- `alpha`: a number between 1 and zero that, if `type="response"` and `family="bynomial"` indicates the splitter to assign the element to the first or the second category of the response variable.

3.2.4. GMERF prediction function

The function presented below gives the possibility to know the predicted values of a new observation through the model obtained using the GMERF function. It is important to highlight the difference with the previous function. The fitted function gives as output the results related to the data used to build the model but it is not possible to predict new observation. On the other hand, the goal of the predict function is to obtain the predicted values of the response variable related to new observations.

```
predict.gmerf(gm, newdata, group, type='response',
              alpha=0.5,predict.all=FALSE,
              re.form=NULL,
              newparam=NULL,terms=NULL, allow.new.levels=TRUE,
              na.action=na.pass,random.only=FALSE)
```

where:

- `object`: is a GMERF object.
- `newdata`: isa data frame in which to look for variables with which to predict. The colums of the dataframe has to be named with the same names of the `dataset` used in the recall of `GMERF` function.
- `type`: is the type of prediction required. The default is on the scale of the linear predictors; the alternative "response" is on the scale of the response variable. Thus for a default binomial model the default predictions are of log-odds (probabilities on logit scale) and type = "response" gives the predicted probabilities. The "terms" option returns a matrix giving the fitted values of each term in the model formula on the linear predictor scale.
- `group`: is a vector or dataframe or matrix object containing the categorical variable

used for the random effects.

For all other inputs refer to the predict function of the stats package [28].

Summary of the GMERF function

The last auxiliary function is useful to have complete view of the output of the algorithm. It is possible to know in the details all the element displayed at the console looking at the documentation of the `R` package base, [27] The generic call of the function is the following

```
summary.gmerf(gmf)
```

where `gmf` is a GMERF .

3.3. Semi-parametric mixed-effects models for a continuous response

Consider the semi parametric method for unsupervised classification presented in the second strand of research. The first function presented is SPEM that can be invoked through the following code:

```
SPEM(case,dataset,resp_var_name,ran_var_name=NULL,
    groups_var_name,iteration=30,iteration_for_sup
    port_reduction=20,iteration_fix=20,
    tollweight=0.02,toll=0.5,toll_ran_eff=10^-3
    ,toll_fix_eff=10^-5,NumMinObs=9)
```

and the input are:

- `case`: a string describing the type of random effects to be considered. It can be equal to `'int','slope','inteslope'`.To further understand the distinctions between the three cases, it is useful to look at the model in Eq.(2.4). In case of `'int'` the model is modified in the following way: $\mathbf{y}_i = \mathbf{X}_i\beta_0+\mathbf{1}c_{0l}+\mathbf{Z}_i\beta_1+\boldsymbol{\epsilon}_i$. In this case the matrix $\mathbf{Z}_i$, that in the initial model contained the random covariates of the model, is treated as a fixed component. Indeed if the choice of the `case` is `'int'` the input `ran_var_name` will not be considered and all the covariates of the dataframe object given as input in `dataset` will be treated as fixed components. Moreover, this is the only case in which it is possible to use the function's default setting for input `ran_var_name`, in fact in all other cases the function will return an error. If the choose is `'slope'` the variable contained in `ran_var_name` will be considered as the random variable of the model and it takes the following form: $\mathbf{y}_i = \mathbf{1}\beta_0 + \mathbf{X}_i\beta_1 + \mathbf{Z}_i c_{1l} + \boldsymbol{\epsilon}_i$. Finally

in the case of `'inteslope'` both slope and intercept are considered and the model takes the following form $\mathbf{y}_i = \mathbf{X}_i\beta + \mathbf{Z}_i c_{1l} + \mathbf{1}c_{0l} + \boldsymbol{\epsilon}_i$

- `dataset`: a dataframe object containing all the statistical units to be considered.
- `resp_var_name`: a string holding the names of the columns in the `dataset` collection that constitutes the model's response variable.
- `ran_var_name`: a string containing the name of the covariates of `dataset` to be considered as the random effect of the model.
- `groups_var_name`: a string with the name of the column of `dataset` to be considered as the one containing information regarding the groups of the various statistical units.
- `iteration`: an integer indicating the maximum number of iteration that the algorithm can perform.
- `iteration_fix`: an integer indicating the maximum number that the algorithm can perform to estimate the fixed component of the model.
- `iteration_for_support_reduction`: an integer indicating the maximum number of iterations the algorithm can perform for the support reduction.
- `tollweight`: a real number indicating the minimum value a weight can take.
- `toll`: a real number indicating the minimum Euclidean distance for two neighboring nodes to be joined into one.
- `toll_ran_eff`:a real number that indicates the minimum difference between $\mathbf{c}_l$ of two consecutive iterations to stop the algorithm.
- `toll_fix_eff`:a real number that indicates the minimum difference between $\boldsymbol{\beta}$'s of two consecutive iterations to stop the algorithm.
- `NumMinObs`: An integer indicating the minimum number of observations a group must have to be considered in the algorithm. The default settings are set to 9 so that all groups with at least 10 observations are considered.

The SPEM function gives as output an object of the SPEM class that consists of the following elements:

- `knots`: is a matrix containing the c_{kl}'s parameter of the model in Eq.(2.4) .It is important to notice that if the model was obtained by giving as input the `'int'` or `'slope' case` the dimension of the matrix are $M \times 1$ matrix where M is the

number of subpopolations identified by the algorithm. While if the chosen case is `'inteslope'` the dimensions will be $M \times 2$.

- `Param`: is a vector containing the fixed parameters of the model, $\boldsymbol{\beta}$.
- `sig`: is the estimated standard deviation of the errors of the model.
- `groups`: is a dataframe object of dimension $N \times 2$ where N is the total number of groups in the model. The first column is a list of all the groups into which the observations were divided. While the second describes the new division into subpopulations.
- `weights`: is a vector of double containing the weights of the discrete distribution of the random parameters.
- `case`: is a string containing the case of the model.
- `VarName`: is a dataframe object containing the name of the variable of the dataset divided in the two components, fixed and random.

3.4. Semi-parametric mixed-effects models for a bivariate continuous response

When the response variable is bivariate, we refer to the BSPEM function that can be invoked with the following code:

```
BSPEM(case,dataset,resp_var_name1,resp_var_name2,
    ran_var_name1=NULL,ran_var_name2=NULL,
    fix_var_name1,fix_var_name2,
    groups_var_name,iteration=30,
    iteration_for_support_reduction=20,
    tollweight=0.05,toll=0.3,
    toll_ran_eff=10^-2,toll_fix_eff=10^-2,
    iteration_fix=20,NumMinObs=9)
```

Due to their absolute equality, the inputs are only stated for the first component:

- `case`:a string describing the type of random effects to be considered. It can be equal to `'int'`,`'slope'`,`'inteslope'`.To further understand the distinctions between the three cases, it is useful to look at the model in Eq.(2.22). In case of `'int'` the model

is modified in the following way:

$$\begin{pmatrix} \mathbf{y}_{1,i} \\ \mathbf{y}_{2,i} \end{pmatrix}^T = \mathbf{1}_{n_i} \begin{pmatrix} c_{1,1m} \\ c_{2,1k} \end{pmatrix}^T + \sum_{p=1}^{P} \mathbf{x}_{ip} \begin{pmatrix} \boldsymbol{\beta}_{1p} \\ \boldsymbol{\beta}_{2p} \end{pmatrix}^T + \mathbf{z}_i \begin{pmatrix} \beta_{1,p+1} \\ \beta_{2,p+1} \end{pmatrix}^T + \begin{pmatrix} \boldsymbol{\epsilon}_{1,i} \\ \boldsymbol{\epsilon}_{2,i} \end{pmatrix}^T$$

In this case the matrix $\mathbf{z}_i$, that in the initial model contained the random covariates of the model, is treated as a fixed component. Indeed if the choice of the `case` is `'int'` the inputs `ran_var_name1` and `ran_var_name2` will be considered as fixed effect, moreover, this is the only case in which it is possible to use the function's default setting for inputs `ran_var_name1` and `ran_var_name2`, in fact in all other cases the function will return an error. If the choice is `'slope'` the variable contained in `ran_var_name1` and `ran_var_name2` will be considered as the random variable of the model and it takes the following form:

$$\begin{pmatrix} \mathbf{y}_{1,i} \\ \mathbf{y}_{2,i} \end{pmatrix}^T = \mathbf{1}_{n_i} \begin{pmatrix} \beta_{1,0} \\ \beta_{2,0} \end{pmatrix}^T + \sum_{p=1}^{P} \mathbf{x}_{ip} \begin{pmatrix} \boldsymbol{\beta}_{1p} \\ \boldsymbol{\beta}_{2p} \end{pmatrix}^T + \mathbf{z}_i \begin{pmatrix} c_{1,2m} \\ c_{2,2k} \end{pmatrix}^T + \begin{pmatrix} \boldsymbol{\epsilon}_{1,i} \\ \boldsymbol{\epsilon}_{2,i} \end{pmatrix}^T$$

Finally in the case of `'inteslope'` both slope and intercept are considered and the model takes the following form:

$$\begin{pmatrix} \mathbf{y}_{1,i} \\ \mathbf{y}_{2,i} \end{pmatrix}^T = \mathbf{1}_{n_i} \begin{pmatrix} c_{1,1m} \\ c_{2,1k} \end{pmatrix}^T + \sum_{p=1}^{P} \mathbf{x}_{ip} \begin{pmatrix} \boldsymbol{\beta}_{1p} \\ \boldsymbol{\beta}_{2p} \end{pmatrix}^T + \mathbf{z}_i \begin{pmatrix} c_{1,2m} \\ c_{2,2k} \end{pmatrix}^T + \begin{pmatrix} \boldsymbol{\epsilon}_{1,i} \\ \boldsymbol{\epsilon}_{2,i} \end{pmatrix}^T$$

`ran_var_name`, in fact in all other cases the function will return an error. If the choose is `'slope'` the variable contained in `ran_var_name` will be considered as the random variable of the model and it takes the following form: $\mathbf{y}_i = \mathbf{1}\beta_0 + \mathbf{X}_i\beta_1 + \mathbf{Z}_i c_{1l} + \boldsymbol{\epsilon}_i$. Finally in the case of `'inteslope'` both slope and intercept are considered and the model takes the following form $\mathbf{y}_i = \mathbf{X}_i\beta + \mathbf{Z}_i c_{1l} + \mathbf{1}c_{0l} + \boldsymbol{\epsilon}_i$

- `dataset`: a dataframe object containing all the statistical units to be considered.
- `resp_var_name1`: a string holding the names of the columns in the `dataset` collection that constitutes the model's response variable.
- `ran_var_name1`: a string containing the name of the covariates of `dataset` to be considered as the random effect of the model.

- `fix_var_name1`: a string containing the name of the covariates of `dataset` to be considered as the fixed effect of the model.
- `groups_var_name`: a string with the name of the column of `dataset` to be considered as the one containing information regarding the groups of the various statistical units.
- `iteration`: an integer indicating the maximum number of iteration that the algorithm can perform.
- `iteration_fix`: an integer indicating the maximum number that the algorithm can perform to estimate the fixed component of the model.
- `iteration_for_support_reduction`: an integer indicating the maximum number of iterations the algorithm can perform for the support reduction.
- `tollweight`: a real number indicating the minimum value a weight can take.
- `toll`: a real number indicating the minimum Euclidean distance for two neighboring nodes to be joined into one.
- `toll_ran_eff`:a real number that indicates the minimum difference between $\mathbf{c}_l$ of two consecutive iterations to stop the algorithm.
- `toll_fix_eff`:a real number that indicates the minimum difference between $\boldsymbol{\beta}$'s of two consecutive iterations to stop the algorithm.
- `NumMinObs`: An integer indicating the minimum number of observations a group must have to be considered in the algorithm. The default settings are set to 9 so that all groups with at least 10 observations are considered.

  The BSPEM function gives as output an object of the BSPEM class composed by the following elements:

  - `knots`:a list of 2 matrices containing the random coefficients of the model in Eq.(2.22). If the model was obtained by giving as input the `'int'` or `'slope' case` case the dimension of the matrices will be $1 \times M$ and $1 \times K$ respectively, where M and K are the numbers of subpopolations identified by the algorithm. In contrast if the chosen case is `'inteslope'` the dimensions of the matrices will be $2 \times M$ and $2 \times K$.
  - `Param`: a list ov vectors containing the fixed parameters of the model.
  - `sig`: is the estimated covariance matrix of the errors of the model.
  - `groups`: a vector of strings containing the new division of the subpopulation.

- `weights`: a list of vectors of double containing the weights of the discrete distribution of the random parameters.
- `case`: a string containing the case of the model.
- `VarName`: a dataframe object containing the name of the variable of the dataset divided in the two components, fixed and random.

3.5. Semi-parametric mixed-effects models for a multinomial response

Finally we present the MSPEM function which refers to the case where the response variable is multinomial. The function can be invoked in the following way:

```
JMSPEM(case,dataset,resp_var_name,ran_var_name=NULL,
    groups_var_name,iteration=30,iteration_for_sup
    port_reduction=20,iteration_fix=20,
    tollweight=0.05,toll=1,toll_ran_eff=10^-2
    ,toll_fix_eff=10^-2,NumMinObs=9)
```

Where:

- `case`: is a string that describes the type of random effects to be considered. It can be equal to `'int','slope','inteslope'`. In case of `'int'` the model is modified in the following way: $\boldsymbol{\eta}_{ik} = \alpha_{1k}\mathbf{x}_{1i} + \alpha_{2k}\mathbf{x}_{2i} + \delta_{ik}$. In this case the matrix $\mathbf{Z}_{1i}$, that in the initial model contained the random covariates of the model, is replaced with the matrix $\mathbf{x}_{2i}$ and therefore treated as a fixed component. Indeed if the choice of the `case` is `'int'` the input `ran_var_name` will not be considered and all the covariates of the dataframe object given as input in `dataset` will be treated as fixed components. Moreover, this is the only case in which it is possible to use the function's default setting for input `ran_var_name`, in fact in all other cases the function will return an error. If the choose is `'slope'` the variable contained in `ran_var_name` will be considered as the random variable of the model and it takes the following form: $\boldsymbol{\eta}_{ik} = \alpha_{1k} + \alpha_{2k}\mathbf{x}_{1i} + \delta_{ik}\mathbf{z}_{1i}$. Finally in the case of `'inteslope'` both slope and intercept are considered and the model takes the following form $\boldsymbol{\eta}_{ik} = \alpha_k\mathbf{x}_{1i} + \delta_{1ik} + \delta_{2ik}\mathbf{z}_{1i}$
- `dataset`: a dataframe object containing all the statistical units to be considered.
- `resp_var_name`: a string holding the names of the columns in the `dataset` collection that constitutes the model's response variable. It is important to note the fact that

the algorithm is built to handle response variables with no more than 5 categories.

- `ran_var_name`: a string containing the name of the covariates of `dataset` to be considered as the random effect of the model.
- `groups_var_name`: a string with the name of the column of `dataset` to be considered as the one containing information regarding the groups of the various statistical units.
- `iteration`: an integer indicating the maximum number of iterations that the algorithm can perform.
- `iteration_fix`: an integer indicating the maximum number of iterations that the algorithm can perform to estimate the fixed component of the model.
- `iteration_for_support_reduction`: an integer indicating the maximum number of iterations the algorithm can perform for the support reduction.
- `tollweight`: a real number indicating the minimum value a weight can take.
- `toll`: a real number indicating the minimum Euclidean distance for two neighboring nodes to be joined into one.
- `toll_ran_eff`:a real number that indicates the minimum difference between random parameters estimated in two consecutive iterations for convergence to have been achieved.
- `toll_fix_eff`:a real number that indicates the minimum difference between $\boldsymbol{\beta}$'s of two consecutive iterations to stop the algorithm.
- `NumMinObs`: An integer indicating the minimum number of observations a group must have to be considered in the algorithm. The default settings are set to 9 so that all groups with at least 10 observations are considered.

The JMSPEM function generates a JMSPEM object containing the following elements:

- `knots`: is a list of K matrices containing the random parameters of the model in Eq.(2.28). The number of columns in the matrices depends on the case chosen, 2 for the `inteslope` case 1 for the other two. While the number of rows is equal to the number of subpopulations found for the relative contrast.
- `Param`: is a vector containing the fixed parameters of the model, δ_{jik}.
- `weights`: is a multidimensional matrix containing all the weights of the model. The number of matrix dimensions depends on the number of contrasts the algorithm has

to perform. While the dimensions are equal to the number of subpopulations found for the relative contrast.

- `weightsMarg`: is a list of vectors of doubles containing the relative weights to the nodes of the discrete distribution of random parameters for each contrast.
- `case`: is a string containing the case of the model.
- `VarName`: is a dataframe object containing the name of the variable of the dataset divided in the two components, fixed and random.

3.6. Auxiliary functions

Associated with the three functions just presented are 3 auxiliary functions useful to the user in studying the output of the models. Since all the functions use as input the output object of the related statistical model, we will refer to the SPEM,BSPEM and JMSPEM object as the output object.

3.6.1. Fitted function

The first function allows to visualize the results of the model intended as a method for unsupervised classification. Indeed it takes as input a vector containing the groups related to one or more new observations and returns a vector of the same length containing the subpopulation related to each unit. The calls of the functions in the three cases are the following:

```
fitted.SPEM(newdata,mod)
fitted.BSPEM(newdata,mod)
fitted.JMSPEM(newdata,mod)
```

where `newdata` is a vector of string containing the group of the new observations and mod is an output object.

3.6.2. Prediction function

The second auxiliary function we present is the predict function that allows to predict new observations. This function takes in input an output object and a dataframe object and gives as output a dataframe object containing the predicted values of the response variable related to each new observations. In particular the call of the function is the following:

```
predict.SPEM(newdata,mod)
predict.BSPEM(newdata,mod)
predict.JMSPEM(newdata,mod)
```

and the input are:

- `newdata`: a dataframe object containing the values of the independent variables whose response is to be predicted. It is important to emphasize the fact that the names of the columns in the dataset must be the same as those given as input to the function used to obtain the output object that will be used.
- `mod`: an output object.

While the output is a vector containing as many elements as the rows of the dataset given as input. The columns of the output are 2 for the `predict.BSPEM` function since the response variable is binomial.

3.6.3. Discrete Mass Plot

The last auxiliary function is a graphical function useful to visualize the random component of the model. In particular the plot obtained through the function is a dotplot containing as many dots as the number of subpopolation found by the algorithm and they represent the parameters of the random components. Specifically, in the case where the SPEM function result is to be displayed, there will be a plot with M dots; in the case of the BSPEM function, there will be a window divided into two plots containing M and K dots, respectively. And finally in the multinomial case there will be K windows, where K is the number of contrasts, containing as many dots as there are subpopulations found. The width of the points is given by the relative weights while the position varies according to the `case` chosen. If the case chosen is `int` or `slope` on the y-axis will be displayed the values related to the intercept or the slope, respectively. In contrast, if the case chosen is `inteslope` on the x-axis there will be the value of the intercept and on the y-axis the one of the slope. The call of the function is the following:

```
DiscreteMassPlot.SPEM(mod)
DiscreteMassPlot.BSPEM(mod)
DiscreteMassPlot.JMSPEM(mod)
```

Where `mod` is an output object.

4 | Examples

This chapter will present examples through which we aim to facilitate understanding of model outputs and auxiliary functions. In addition, attention will be paid to some of the input parameters of the functions to explain both possible critical issues of the algorithms and how to make the best use of its potentials. The datasets used for the examples have two different natures depending on the branch of research they refer to. In the first case, the dataset is inspired by the one presented in articles [6] and [24]; in the second case, it is simulated. Also in this second case, inspiration was taken from the articles [16], [17] and [19] but the simulations have slight differences and still do not refer to real cases. For ease of understanding and to facilitate synthesis in the section below are presented the simulations from which the datasets originated and the command lines needed to load them to the workspace.

4.1. Data

The datasets used in the example related to the GMET and GMERF functions are inspired to the ones presented in [24] and [6] and even if they are really similar they are treated individually because there are some differences that it is good to point out. It is important to notice that the all the datasets are ready to be used since the whole pre-processing work has already been done. Finally all the lines of command used to load the data in the workspace use the `data()` function from the `R` package `utils`, [29].

4.1.1. GMET dataset

The first dataset we present consists of 18612 rows and 10 columns. The statistical units of the model are the students, the response variable Y is the career `status`, a two-level factor we code as a binary variable:

- `status` $= 1$ for careers definitely completed with graduation;
- `status` $= 0$ for careers definitely concluded with a dropout.

Instead, the categorical variable used to divide students into the 19 groups is the course of study contained in the "Degree.Program.in" column. Finally all the other covariates of the dataset that in the example will be used as fixed effects are described in Table.4.1. It is possible to load the dataset into the workspace using the command

Variable	Description	Type of variable
`Sex`	gender	factor (2 levels: M, F)
`Nationality`	nationality	factor (Italian, foreigner)
`PreviousStudies`	high school studies	factor (*Liceo Scientifico*, *Istituto Tecnico*, Other)
`AdmissionScore`	PoliMi admission test result	real number
`AccessToStudiesAge`	age at the beginning of the BSc studies at PoliMi	natural number
`WeightedAvgEval1.1`	weighted average of the evaluations during the first semester of the first year	real number
`AvgAttempts1.1`	average number of attempts to be evaluated on subjects during the first semester of the first year (passed and failed exams)	real number
`TotalCredits1.1`	number of ECTS credits obtained by the student during the first semester of the first year	natural number

Table 4.1: List and explanation of variables at student level to be included as covariates in the GMET model

```
data(GMET.data,package="SpMEMs")
```

4.1.2. GMERF dataset

The dataset useful for testing the GMERF function has 24736 rows and 8 columns. Again, the statistical units are students divided into 19 courses of study. The response variable and the variable allowing division into groups remain unchanged. All information regarding the other covariates that will be used as fixed components in the example is explained in the Table.4.2

Finally the command line to load the data is the following:

```
data(GMERF.data,package="SpMEMs")
```

Variable	Description	Type of variable
`Sex`	gender	factor (2 levels: M, F)
`Nationality`	nationality	factor (Italian, foreigner)
`PreviousStudies`	high school studies	factor (*Liceo Scientifico*, *Istituto Tecnico*, Other)
`WeiAvgEval1.1`	weighted average of the evaluations during the first semester of the first year	real number
`AvgAtt1.1`	average number of attempts to be evaluated on subjects during the first semester of the first year (passed and failed exams)	real number
`TotalCredits1.1`	number of ECTS credits obtained by the student during the first semester of the first year	natural number
`Degree.Programme.in`	Degree program the student is enrolled in (grouping variable)	factor

Table 4.2: List and explanation of variables at student level to be included as covariates in the GMERF model

4.1.3. SPEM datasets

The dataset available to test the SPEM function was created following the same DGP as the one in article [16] changing only a few values. It is good to specify that for each of the 3 cases , namely `int slope` and `inteslope`, a slightly different dataset was simulated. There are two factors common to all 3 datasets, the first being the simulation of the fixed and random effects observations that are sampled from the following distribution:

$$\begin{aligned} \mathbf{x}_1 &\sim \mathcal{N}(0.30, 0.16), & \mathbf{z}_1 &\sim \mathcal{N}(0.1, 0.16), \\ \mathbf{x}_2 &\sim \mathcal{N}(0.28, 0.16), & \mathbf{z}_2 &\sim \mathcal{N}(0.12, 0.16), \\ \mathbf{x}_3 &\sim \mathcal{N}(0.27, 0.16), & \mathbf{z}_3 &\sim \mathcal{N}(0.8, 0.16), \end{aligned} \tag{4.1}$$

The second factor is the composition of the dataset, indeed the datasets are composed by 855 observation nested within 9 groups. The first 3 groups have 100 observations the second 3 have 90 observations and the last triplet have 95 observations each. Below all datasets are explained by specifying which case they refer to:

- `inteslope`

$$\begin{cases} \mathbf{y}_i = 3\mathbf{x}_1 + 5 + 10\mathbf{z}_1 + \boldsymbol{\epsilon}_i & i = 1,2,3 \\ \mathbf{y}_i = 3\mathbf{x}_2 + 2 + 5\mathbf{z}_2 + \boldsymbol{\epsilon}_i & i = 4,5,6 \\ \mathbf{y}_i = 3\mathbf{x}_3 + 0 + 2\mathbf{z}_3 + \boldsymbol{\epsilon}_i & i = 7,8,9 \end{cases} \tag{4.2}$$

- `int`

$$\begin{cases} \mathbf{y}_i = 10\mathbf{x}_1 + 5 + 2\mathbf{z}_1 + \boldsymbol{\epsilon}_i & i = 1,2,3 \\ \mathbf{y}_i = 10\mathbf{x}_2 + 2 + 2\mathbf{z}_2 + \boldsymbol{\epsilon}_i & i = 4,5,6 \\ \mathbf{y}_i = 10\mathbf{x}_3 + 0 + 2\mathbf{z}_3 + \boldsymbol{\epsilon}_i & i = 7,8,9 \end{cases} \tag{4.3}$$

- `slope`

$$\begin{cases} \mathbf{y}_i = 3\mathbf{x}_1 + 10\mathbf{z}_1 + \boldsymbol{\epsilon}_i & i = 1,2,3 \\ \mathbf{y}_i = 3\mathbf{x}_2 + 5\mathbf{z}_2 + \boldsymbol{\epsilon}_i & i = 4,5,6 \\ \mathbf{y}_i = 3\mathbf{x}_3 + 1\mathbf{z}_3 + \boldsymbol{\epsilon}_i & i = 7,8,9 \end{cases} \tag{4.4}$$

Once the data were simulated, the final dataset was constructed by merging previous sampling in the manner shown in Eq.(4.5)

$$\begin{array}{cccc} y & x & z & groups \\ \mathbf{y}_{11} & \mathbf{x}_{11} & \mathbf{z}_{11} & \text{"1"} \\ \mathbf{y}_{21} & \mathbf{x}_{11} & \mathbf{z}_{11} & \text{"1"} \\ \dots & \dots & \dots & \\ \mathbf{y}_{100\ 1} & \mathbf{x}_{100\ 1} & \mathbf{z}_{100\ 1} & \text{"1"} \\ \mathbf{y}_{12} & \mathbf{x}_{12} & \mathbf{z}_{12} & \text{"2"} \\ \dots & \dots & \dots & \\ \mathbf{y}_{95\ 9} & \mathbf{x}_{95\ 9} & \mathbf{z}_{95\ 9} & \text{"9"} \end{array} \tag{4.5}$$

Finally it is possible to load the datasets in the worksspace throgh the following line of commands:

```
data(SPEM.inteslope.data,package="SpMEMs")
data(SPEM.int.data,package="SpMEMs")
data(SPEM.slope.data,package="SpMEMs")
```

4.1.4. BSPEM datasets

Now let us consider the datasets used to create the examples related to the BSPEM function, again we can find similarities with the DGP from Article [17]. We also find,

again, three slightly different datasets, one for each instance of the input case. The three datasets have the same composition and the observations for the fixed and random components are the same but the models are different. The distributions from which the fixed and random observations were drawn are presented below:

$$\begin{aligned} z_i &\sim \mathcal{N}(0.1, 0.16) \quad i = 1, 2, 3 \\ z_i &\sim \mathcal{N}(0.12, 0.16) \quad i = 4, 5, 6 \\ z_i &\sim \mathcal{N}(0.8, 0.16) \quad i = 7, 8, 9 \end{aligned} \tag{4.6}$$

$$\begin{aligned} x_i &\sim \mathcal{N}(0.3, 0.16) \quad i = 1, 2, 3 \\ x_i &\sim \mathcal{N}(0.28, 0.16) \quad i = 4, 5, 6 \\ x_i &\sim \mathcal{N}(0.27, 0.16) \quad i = 7, 8, 9 \end{aligned} \tag{4.7}$$

and

$$\boldsymbol{\epsilon}_i \sim \mathcal{N}_2\Big(\mathbf{0}, \boldsymbol{\Sigma} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}\Big) \quad i = 1, \ldots, 9. \tag{4.8}$$

While the composition of each of the three models is the following:

- `inteslope`

$$\begin{gathered} \begin{pmatrix} \mathbf{y}_{1,i} \\ \mathbf{y}_{2,i} \end{pmatrix}^T = \mathbf{1}_{n_i} \begin{pmatrix} c_{1,1m} \\ c_{2,1k} \end{pmatrix}^T + \mathbf{x}_i \begin{pmatrix} \beta_1 \\ \beta_2 \end{pmatrix}^T + \mathbf{z}_i \begin{pmatrix} c_{1,2m} \\ c_{2,2k} \end{pmatrix}^T + \boldsymbol{\epsilon}_i \\ i = 1, \ldots, 9 \qquad m = 1, \ldots, M \qquad k = 1, \ldots, K \\ \boldsymbol{\epsilon}_i^T = \begin{pmatrix} \boldsymbol{\epsilon}_{1,i} \\ \boldsymbol{\epsilon}_{2,i} \end{pmatrix} \sim \mathcal{N}_2(\mathbf{0}, \boldsymbol{\Sigma}) \qquad ind. \end{gathered} \tag{4.9}$$

  and the value of the parameter can be consulted in Table.4.3.

- `int`

$$\begin{gathered} \begin{pmatrix} \mathbf{y}_{1,i} \\ \mathbf{y}_{2,i} \end{pmatrix}^T = \mathbf{1}_{n_i} \begin{pmatrix} c_{1,1m} \\ c_{2,1k} \end{pmatrix}^T + \mathbf{x}_i \begin{pmatrix} \beta_{1,1} \\ \beta_{1,2} \end{pmatrix}^T + \mathbf{z}_i \begin{pmatrix} \beta_{1,2} \\ \beta_{2,2} \end{pmatrix}^T + \boldsymbol{\epsilon}_i \\ i = 1, \ldots, 9 \qquad m = 1, \ldots, M \qquad k = 1, \ldots, K \\ \boldsymbol{\epsilon}_i^T = \begin{pmatrix} \boldsymbol{\epsilon}_{1,i} \\ \boldsymbol{\epsilon}_{2,i} \end{pmatrix} \sim \mathcal{N}_2(\mathbf{0}, \boldsymbol{\Sigma}) \qquad ind. \end{gathered} \tag{4.10}$$

  and the value of the parameter can be consulted in Table.4.4.

- `slope`

$$\begin{pmatrix} \mathbf{y}_{1,i} \\ \mathbf{y}_{2,i} \end{pmatrix}^T = \mathbf{1}_{n_i} \begin{pmatrix} \beta_{0,1} \\ \beta_{0,2} \end{pmatrix}^T + \mathbf{x}_i \begin{pmatrix} \beta_{1,1} \\ \beta_{1,2} \end{pmatrix}^T + \mathbf{z}_i \begin{pmatrix} c_{1,2m} \\ c_{2,2k} \end{pmatrix}^T + \boldsymbol{\epsilon}_i$$
$$i = 1, \dots, 9 \qquad m = 1, \dots, M \qquad k = 1, \dots, K \tag{4.11}$$
$$\boldsymbol{\epsilon}_i^T = \begin{pmatrix} \boldsymbol{\epsilon}_{1,i} \\ \boldsymbol{\epsilon}_{2,i} \end{pmatrix} \sim \mathcal{N}_2(\mathbf{0}, \boldsymbol{\Sigma}) \qquad ind.$$

and the value of the parameter can be consulted in Table.4.5.

	First response parameters	Second response parameters
$i = 1, 2, 3$	$c_{1,11} = 5$ $c_{1,21} = 10$ $\beta_1 = 3$	$c_{2,11} = 50$ $c_{2,21} = 100$ $\beta_2 = 30$
$i = 4, 5, 6$	$c_{1,12} = 2$ $c_{1,22} = 5$ $\beta_1 = 3$	$c_{2,11} = 20$ $c_{2,21} = 50$ $\beta_2 = 30$
$i = 7, 8, 9$	$c_{1,13} = 0$ $c_{1,23} = -2$ $\beta_1 = 3$	$c_{2,12} = 10$ $c_{2,22} = 20$ $\beta_2 = 30$

Table 4.3: Set of parameters used to simulate the response variable following the model in Eq.(4.9). The intercepts and the coefficients of $\mathbf{z}$ differ across subpopulations, while the coefficients β of x are fixed. We impose a structure with three subpopulations in both the responses (M=3,K=3).

	First response parameters	Second response parameters
$i = 1, 2, 3$	$c_{1,11} = 5$ $\beta_{1,1} = 10$ $\beta_{1,2} = 3$	$c_{2,11} = 50$ $\beta_{2,1} = 10$ $\beta_{2,2} = 3$
$i = 4, 5, 6$	$c_{1,12} = 2$ $\beta_{1,1} = 10$ $\beta_{1,2} = 3$	$c_{2,11} = 50$ $\beta_{2,1} = 10$ $\beta_{2,2} = 3$
$i = 7, 8, 9$	$c_{1,13} = -2$ $\beta_{1,1} = 10$ $\beta_{1,2} = 3$	$c_{2,12} = 20$ $\beta_{2,1} = 10$ $\beta_{2,2} = 3$

Table 4.4: Set of parameters used in Eq. (4.10) to simulate data. We impose a structure with three subpopulations in the first response (M=3) and two subpopulations in the second one (K=2).

Once the data were simulated, the final dataset was constructed by merging previous sampling in the manner shown in Eq.(4.12)

	First response parameters	Second response parameters
$i = 1, 2, 3$	$c_{1,11} = 10$ $\beta_{0,1} = 0$ $\beta_{1,1} = 3$	$c_{2,11} = 1$ $\beta_{0,2} = 0$ $\beta_{1,2} = 3$
$i = 4, 5, 6$	$c_{1,12} = 5$ $\beta_{0,1} = 0$ $\beta_{1,1} = 3$	$c_{2,11} = 1$ $\beta_{0,2} = 0$ $\beta_{1,2} = 3$
$i = 7, 8, 9$	$c_{1,13} = -2$ $\beta_{0,1} = 0$ $\beta_{1,1} = 7$	$c_{2,12} = -3$ $\beta_{0,2} = 0$ $\beta_{1,2} = 3$

Table 4.5: Set of parameters used in Eq. (4.11) to simulate data. We impose a structure with three subpopulations in the first response (M=3) and two subpopulations in the second one (K=2).

$$
\begin{array}{cccccc|c}
y1 & y2 & x1 & x1 & z1 & z2 & groups \\
\mathbf{y1}_{1,1} & \mathbf{y2}_{1,1} & \mathbf{x1}_{1,1} & \mathbf{x2}_{1,1} & \mathbf{z1}_{1,1} & \mathbf{z2}_{1,1} & \text{"1"} \\
\mathbf{y1}_{2,1} & \mathbf{y2}_{2,1} & \mathbf{x1}_{2,1} & \mathbf{x2}_{2,1} & \mathbf{z1}_{2,1} & \mathbf{z2}_{2,1} & \text{"1"} \\
\dots & \dots & \dots & \dots & \dots & \dots & \\
\mathbf{y1}_{100,1} & \mathbf{y2}_{100,1} & \mathbf{x1}_{100,1} & \mathbf{x2}_{100,1} & \mathbf{z1}_{100,1} & \mathbf{z2}_{100,1} & \text{"1"} \\
\dots & \dots & \dots & \dots & \dots & \dots & \\
\mathbf{y1}_{1,2} & \mathbf{y2}_{1,2} & \mathbf{x1}_{1,2} & \mathbf{x2}_{1,2} & \mathbf{z1}_{1,2} & \mathbf{z2}_{1,2} & \text{"2"} \\
\dots & \dots & \dots & \dots & \dots & \dots & \\
\mathbf{y1}_{95,9} & \mathbf{y2}_{95,9} & \mathbf{x1}_{95,9} & \mathbf{x2}_{95,9} & \mathbf{z1}_{95,9} & \mathbf{z2}_{95,9} & \text{"9"}
\end{array}
\tag{4.12}
$$

Finally it is possible to load the datasets in the workspace with the following line of command:

```
data(BSPEM.inteslope.data,package="SpMEMs")
data(BSPEM.int.data,package="SpMEMs")
data(BSPEM.slope.data,package="SpMEMs")
```

4.1.5. JMSPEM datasets

To conclude this first part of the chapter, let us consider the JMPSEM function that is useful if the response variable is a multinomial variable. This last DGP is slightly different from the previous ones, but still very similar to the one in Article [19]. Again the simulated datasets will be three as many as the possible input parameter `case` options. There are 945 observations divided into 10 groups. The first 3 groups have 100 observations each,

the next 4 groups have 90 observations each and the last 3 groups count 95 observations each. The response variable consists of three different categories which implies 2 contrasts therefore two $\boldsymbol{\eta}$ are simulated following the models given below.

(i) Random intercept case ($\boldsymbol{\eta}_{ik} = \alpha_{1k}\mathbf{x}_{1i} + \alpha_{2k}\mathbf{x}_{2i} + \delta_{ik}$)

$$\begin{aligned}
\boldsymbol{\eta}_{i2} &= \begin{cases} +4\mathbf{x}_{1i} - 3\mathbf{x}_{2i} - 7 & i = 1,2,3 \\ +4\mathbf{x}_{1i} - 3\mathbf{x}_{2i} - 4 & i = 4,5,6,7 \\ +4\mathbf{x}_{1i} - 3\mathbf{x}_{2i} - 2 & i = 8,9,10 \end{cases} \\
\boldsymbol{\eta}_{i3} &= \begin{cases} -2\mathbf{x}_{1i} + 2\mathbf{x}_{2i} - 5 & i = 1,\ldots,7 \\ -2\mathbf{x}_{1i} + 2\mathbf{x}_{2i} - 2 & i = 8,9,10 \end{cases}
\end{aligned} \tag{4.13}$$

(ii) Random slope case ($\boldsymbol{\eta}_{ik} = \alpha_{1k} + \alpha_{2k}\mathbf{x}_{1i} + \delta_{ik}\mathbf{z}_i$)

$$\begin{aligned}
\boldsymbol{\eta}_{i2} &= \begin{cases} -1 - 3\mathbf{x}_{1i} + 5\mathbf{z}_i & i = 1,2,3 \\ -1 - 3\mathbf{x}_{1i} + 2\mathbf{z}_i & i = 4,5,6,7 \\ -1 - 3\mathbf{x}_{1i} - 1\mathbf{z}_i & i = 8,9,10 \end{cases} \\
\boldsymbol{\eta}_{i3} &= \begin{cases} -2 + 2\mathbf{x}_{1i} - 2\mathbf{z}_i & i = 1,\ldots.7 \\ -2 + 2\mathbf{x}_{1i} - 6\mathbf{z}_i & i = 8,9,10 \end{cases}
\end{aligned} \tag{4.14}$$

(iii) Random intercept and slope case ($\boldsymbol{\eta}_{ik} = \alpha_k\mathbf{x}_{1i} + \delta_{1ik} + \delta_{2ik}\mathbf{z}_i$)

$$\begin{aligned}
\boldsymbol{\eta}_{i2} &= \begin{cases} -5\mathbf{x}_{1i} - 6 + 5\mathbf{z}_i & i = 1,2,3 \\ -5\mathbf{x}_{1i} - 4 + 2\mathbf{z}_i & i = 4,5,6,7 \\ -5\mathbf{x}_{1i} - 8 - 1\mathbf{z}_i & i = 8,9,10 \end{cases} \\
\boldsymbol{\eta}_{i3} &= \begin{cases} +2\mathbf{x}_{1i} + 1 - 4\mathbf{z}_i & i = 1,\ldots,7 \\ +2\mathbf{x}_{1i} - 1 + 2\mathbf{z}_i & i = 8,9,10 \end{cases}
\end{aligned} \tag{4.15}$$

Variables $\mathbf{x}_1$, $\mathbf{x}_2$ and $\mathbf{z}$ are normally distributed with mean equal to 0 and standard deviation equal to 1. Then the relative π_{ijk} of each category of the response variable y are calculated following the formula given in Eq.(2.24) and the category k is assigned if relative π_{ijk} is the highest. Finally, the simulated data with the corresponding response variable are merged, differentially for the `inteslope, slope` and `int` cases, to form the

final result. `inteslope, slope`:

$$\begin{array}{cccc} y & x_1 & z & groups \\ \mathbf{y}_{11} & \mathbf{x}_{11} & \mathbf{z}_{11} & "1" \\ \mathbf{y}_{21} & \mathbf{x}_{11} & \mathbf{z}_{11} & "1" \\ \dots & \dots & \dots & \\ \mathbf{y}_{100\ 1} & \mathbf{x}_{100\ 1} & \mathbf{z}_{100\ 1} & "1" \\ \mathbf{y}_{12} & \mathbf{x}_{12} & \mathbf{z}_{12} & "2" \\ \dots & \dots & \dots & \\ \mathbf{y}_{95\ 10} & \mathbf{x}_{95\ 10} & \mathbf{z}_{95\ 10} & "10" \end{array} \tag{4.16}$$

`int`:

$$\begin{array}{cccc} y & x_1 & x_2 & groups \\ \mathbf{y}_{11} & \mathbf{x_1}_{11} & \mathbf{x_2}_{11} & "1" \\ \mathbf{y}_{21} & \mathbf{x_1}_{11} & \mathbf{x_2}_{11} & "1" \\ \dots & \dots & \dots & \\ \mathbf{y}_{100\ 1} & \mathbf{x_1}_{100\ 1} & \mathbf{x_2}_{100\ 1} & "1" \\ \mathbf{y}_{12} & \mathbf{x_1}_{12} & \mathbf{x_2}_{12} & "2" \\ \dots & \dots & \dots & \\ \mathbf{y}_{95\ 10} & \mathbf{x_1}_{95\ 10} & \mathbf{x_2}_{95\ 10} & "10" \end{array} \tag{4.17}$$

Finally it is possible to load on the workspace the datasets with the following line of command:

```
data(JMSPEM.inteslope.data,package="SpMEMs")
data(JMSPEM.int.data,package="SpMEMs")
data(JMSPEM.slope.data,package="SpMEMs")
```

4.2. Examples

4.2.1. GMET example

The first of the examples presented proposes the analysis of the GMET.dataset using the GMET function. We first load the dataset into the workspace via the previously presented commands. Then we give as input to the GMET function the formula considering "Status" as the response variable , "Degree.Name.out" as the random term and the remaining ones as fixed variables.

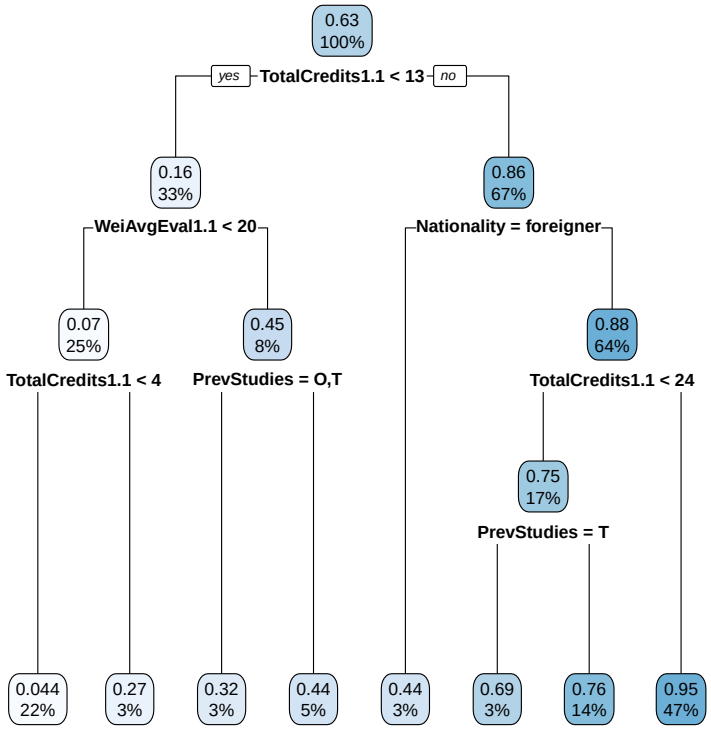


Figure 4.1: The RPART object obtained from the algorithm used to estimate the fixed component function of the model

```
df=data(GMET.data,package="SpMEMs")
gmt <- GMET(formula = Status ~ Sex +Nationality+PrevStudies +
                  AdmissionScore +AccessAge +WeiAvgEval1.1
                     +AvgAtt11 +TotalCredits1.1,dataset = df,
                     random = ~ (1|DegreeName.out))
```

By doing so we obtain a GMET object through which the outputs of some of the auxiliary functions can be shown. We use the `plot.mymixedtree` function to display the RPART obtained by the algorithm to estimate the $\boldsymbol{f}(X_i)$ function of the model in Eq.(2.2). The code is as follows:

```
plot.mymixedtree(gmt)
```

And it plots the graph in Fig.4.1. Each node reports the percentage of observations belonging to the node (second line of the node) and the estimated probability that responses relative to these observations are equal to 1 (first line of the node). Regarding the splitting criteria, left branches correspond to the case in which the condition is satisfied, while right branches correspond to the complementary case. It is also possible to get graphical feedback of the random component of the model. In fact, the `dotplot.GMET` function

allows the display of a dotplot containing estimates of the random intercepts along with a confidence interval of 95%. Also in showing the example the input `rand.int` will be set to TRUE so that the values of the estimated intercepts will be shown in the console. The call of the function is the following:

```
dotplot.GMET(gmt,rand.int=TRUE)
```

and we obtain the plot in Fig4.2 and the following lines on the console:

```
    $DegreeName.out

$DegreeName.out
           (Intercept)
DegPro_1   -0.10806958
DegPro_2    0.36280643
DegPro_3    0.11735336
DegPro_4   -0.39717633
DegPro_5   -0.39812985
DegPro_6    0.52893174
DegPro_7   -0.63827252
DegPro_8   -0.62426716
DegPro_9   -0.07346035
DegPro_10  -0.13081838
DegPro_11  -0.52540271
DegPro_12   1.13881126
DegPro_13   0.49686298
DegPro_14   0.46896147
DegPro_15  -0.08721831
DegPro_16   0.05331538
DegPro_17  -0.21046631
DegPro_18   0.21469630
DegPro_19  -0.21130454
```

4.2.2. GMERF example

Consider now the second function of the first research branch, GMERF. The dataset GMERF.data was loaded into the workspace using the above commands. The columns were divided into the various components and then given as input to the function in the following way:

```
df=data(GMERF.data , package="SpMEMs")
```

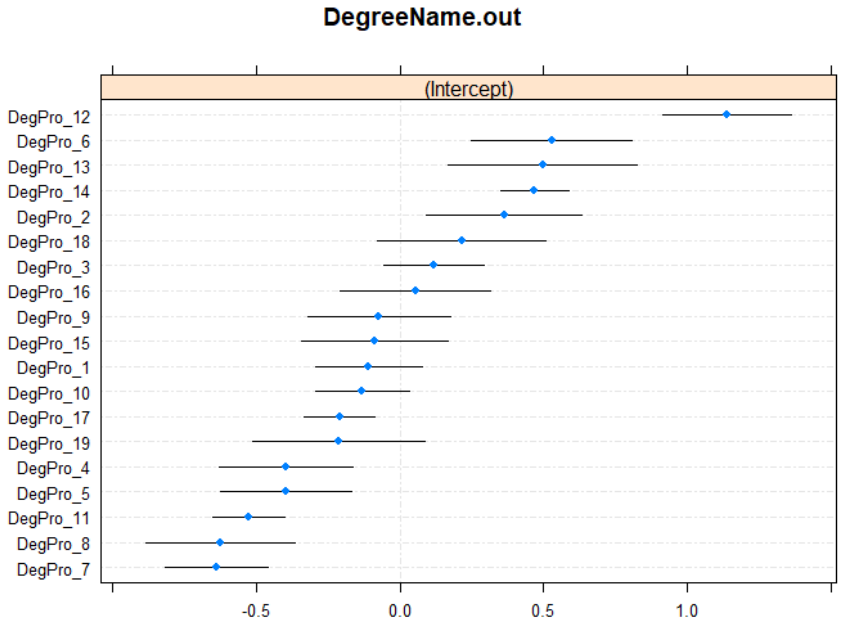


Figure 4.2: For each Engineering programme, the blue dot and the horizontal line marks the estimate and the 95% confidence interval of the corresponding random intercept

```
VarResp=df$Status

FixCov=df[,c("Sex","Nationality","PreviousStudies",
"WeiAvgEval1.1","AvgAtt1.1","TotalCredits1.1")]

groups=factor(df$DegreeProgramme.in)

gmf=GMERF(var_resp_gmerf,cov_gmerf,gruppi_gmerf)
```

These lines of code give as output an object of class GMERF that can be used as input for auxiliary functions. We use the function `summary.GMERF` to get a complete picture of the model's output, via the following command:

```
summary.GMERF(gmf)
```

obtaining as output the following:

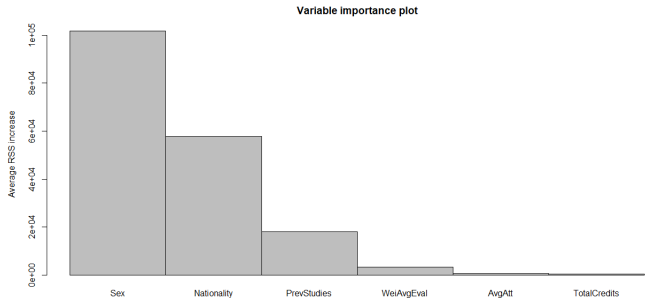
```
    [1] "Mixed␣effects␣model"
Generalized linear mixed model fit by maximum likelihood (Laplace
   Approximation) ['glmerMod']
 Family: binomial  ( logit )
```

```
Formula: y ~ (1 | group)
   Data: glmer.data
 Offset: f.x_ij

     AIC      BIC   logLik deviance df.resid
 12946.3  12962.5  -6471.1  12942.3    24734

Scaled residuals:
    Min      1Q  Median      3Q     Max
-487.57   -0.11    0.17    0.30   12.91

Random effects:
 Groups Name        Variance Std.Dev.
 group  (Intercept) 1.055    1.027
Number of obs: 24736, groups:  group, 19

Fixed effects:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -0.01197    0.23720   -0.05     0.96
[1] "Converged after 9 iterations"
```

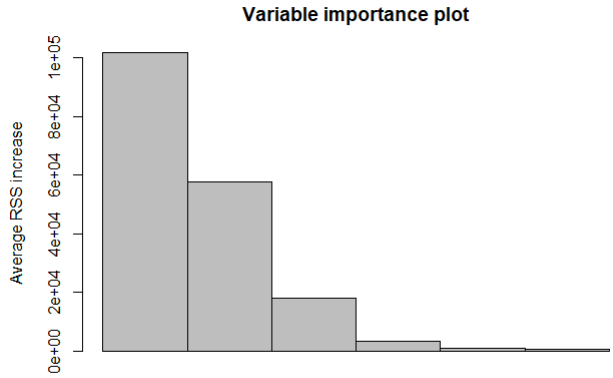
Next, we begin the graphical analysis of the model starting with the VIP related to the random forest used to estimate the fixed component of the model. This graphical tool allows us to understand which variable has greater weight in predicting the response variable. The call of the function is the following:

```
VIP.GMEF(gmf,covnames=c("Sex","Nationality","PrevStudies",
    "WeiAvgEval","AvgAtt","TotalCredits"))
```

and we obtain the plot in Fig.4.3a. The plot in Fig.4.3b is useful to show what happens if the default settings of the input `covnames` are left, in particular you can see the lack of labels on the x-axis.



(a) Variable importance plot without `covnames` default option



(b) Variable importance plot with `covnames` default option

Figure 4.3: VIP showing the importance of the variable in the prediction of the response. The height of the bar is the increase of the residual sum of squares (RSS) when the values of the corresponding variable are randomly permuted

Finally, the graphical function `dotplot.GMERF` can be used to display the random intercept estimated by the model along with a 95 percent confidence interval. The command is as follows:

```
dotplot.GMERF(gmf,rand.int=TRUE)
```

and we obtain the plot in Fig.4.4. Since the input `rand.int` is set on TRUE the following will be displayed:

```
    $group
          (Intercept)
DegPro_1   0.32954564
DegPro_2  -0.03897417
DegPro_3  -0.76485897
```

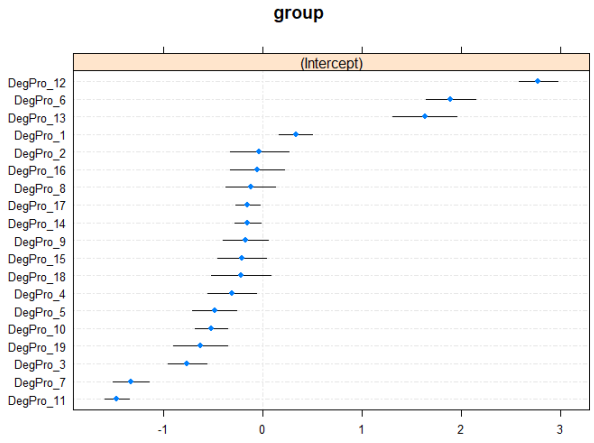


Figure 4.4: Random intercepts relative to the degree programmes estimated by the GMERF model with their confidence intervals.

```
DegPro_4   -0.31434808
DegPro_5   -0.48828902
DegPro_6    1.89198922
DegPro_7   -1.33198113
DegPro_8   -0.12593889
DegPro_9   -0.17658030
DegPro_10  -0.52295826
DegPro_11  -1.47557781
DegPro_12   2.77339301
DegPro_13   1.63024023
DegPro_14  -0.15466802
DegPro_15  -0.21278833
DegPro_16  -0.05719709
DegPro_17  -0.15444244
DegPro_18  -0.22216773
DegPro_19  -0.62917500
```

4.2.3. SPEM example

Consider the SPEM function related to the second line of research. The example for this function has a section on the input parameter `toll` that is useful in explaining any problems behind the setting of this parameter. Begin as usual by importing the three datasets simulated in the previous section, then they are given as input to the function. The codes for this first part divided for the three cases are given below:

```
df.inteslope=data(SPEM.inteslope.dataset,package="SpMEMs")
df.int=data(SPEM.int.dataset,package="SpMEMs")
df.slope=data(SPEM.slope.dataset,package="SpMEMs")

smpIs=SPEM(case='inteslope',dataset=df,resp_var_name="y",ran_var_name="z",g
smpI=SPEM(case='int',dataset=df,resp_var_name="y",groups="groups")
smpS=SPEM(case='slope',dataset=df,resp_var_name="y",ran_var_name="z",groups
```

Thus obtaining three SPEM objects that can be used as inputs in auxiliary functions. A first assessment of the work done by the algorithm is possible by looking at the estimated parameters via output visualization. Therefore through the following code:

```
print("Knots␣inteslope")
spmIs$knots
print("Fixed␣params␣inteslope")
spmIs$param

print("Knots␣int")
spmI$knots
print("Fixed␣params␣int")
spmI$param

print("Knots␣slope")
spmS$knots
print("Fixed␣params␣␣␣slope")
spmS$param
```

we obtain:

```
    Knots inteslope
         [,1]      [,2]
[1,] 10.03691  5.056259
[2,] 5.131829  2.182689
[3,] 2.092389  0.157803
    Fixed param inteslope
[1] 3.137923

    Knots int
         [,1]
[1,] 5.149303
[2,] 2.182922
```

```
[3,] 0.018273
    Fixed param int
[1] 1.993742  10.002846

    Knots slope
        [,1]
[1,] 10.033149
[2,] 4.980654
[3,] 0.891233
    Fixed param slope
[1] 3.92342  0.002846
```

The algorithm was applied on simulated data, therefore it is possible to see the successful outcome by contrasting the values in the previous table with those in equations 4.2, 4.3 and 4.4. It is crucial to note that the toll option was left empty, hence the default parameter of 0.5 was utilized to get the results shown above. 50 simulations were performed, with the toll parameter increasing by 0.1 each time from 0 to 5, to demonstrate how the algorithm's findings change as the parameter varies. The number of subpopulations discovered in response to the tolerance parameter's value is shown in Figure 4.5.We can see that the algorithm has issues at the two extremes; in fact, if toll is set to 0, there will be no support reduction and the number of subpopulations will remain the same. In contrast, a tolerance value of 5 would cause all nodes to collapse into one, resulting in an error.It should be noted that the example given here is for simulated data and therefore the choice of parameter is easy since the number of subpopulations is known a priori but in a real case this is not possible. It is therefore recommended to try various values of the tolerance parameter and choose the one that is believed to be most valid based on a preliminary analysis of the data.

To conclude the section on examples related to the spem function, we present a method of using the function dotplot.SPEM. By running the code below it is possible to get a graphical feedback of the random parameter values estimated by the algorithm for all three cases of the parameter case.

```
dotplot.SPEM(spmIs)
dotplot.SPEM(spmI)
dotplot.SPEM(spmS)
```

The result is the plot presented in Fig. 4.6. To be precise, the first quadrant of the plot refers to the `inteslope` case, the second to the `int` case, and the last to the `slope` case. It is worth noting that the three plot points have a same magnitude because the three

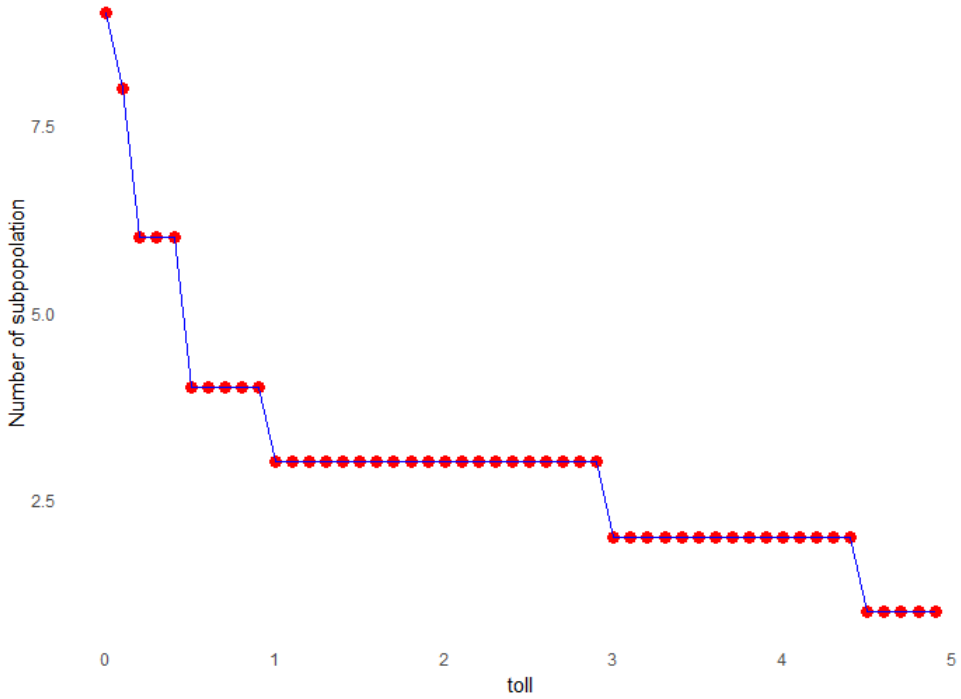


Figure 4.5: Plot of the identified subpopolation in relation to change with the toll parameter.

relative weights have the same value as can be seen with the following code:

```
> spmIs$weights
[1] 0.3333333 0.3333333 0.3333333

> spmI$weights
[1] 0.3333333 0.3333333 0.3333333

> spmS$weights
[1] 0.3333333 0.3333333 0.3333333
```

4.2.4. BSPEM examples

Following the structure of the previous section, we start the work on the BSPEM function examples by loading the previously simulated data into the workspace and then give it as input to the BSPEM function to obtain three BSPEM objects through the following code:

```
df.inteslope=data(BSPEM.inteslope.dataset,package="SpMEMs")
df.int=data(BSPEM.int.dataset,package="SpMEMs")
```

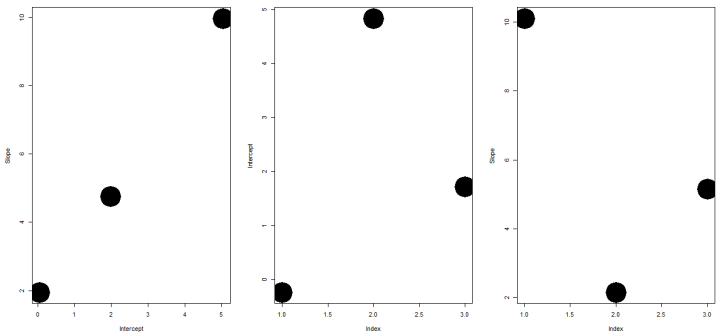


Figure 4.6: Dotplot of the random discrete distributions related to the 3 models in equations 4.2, 4.3 and 4.4. The positions of the dots are the $\mathbf{c}_k l$'s parameters while the diameter of the dot is the weight.

```
df.slope=data(BSPEM.slope.dataset,package="SpMEMs")

bsmpIs=BSPEM(case='inteslope',dataset=df,resp_var_name1="y1",resp_var_
bsmpI=BSPEM(case='int',dataset=df,resp_var_name1="y1",resp_var_name2="
bsmpS=BSPEM(case='slope',dataset=df,resp_var_name1="y1",resp_var_name2=
```

Having obtained the three BSPEM objects, it is possible to check the work done by the algorithm by looking at the outputs `knots` and `param`

```
print("Knots␣inteslope")
bspmIs$knots
print("Fixed␣params␣inteslope")
bspmIs$param

print("Knots␣int")
bspmI$knots
print("Fixed␣params␣int")
bspmI$param

print("Knots␣slope")
bspmS$knots
print("Fixed␣params␣␣slope")
bspmS$param
```

we obtain:

```
    Knots inteslope
[[1]]
```

```
          [,1]         [,2]
[1,] 10.018411   5.120146
[2,]  5.060082   2.074978
[3,] -1.974125 -0.099550

[[2]]
          [,1]       [,2]
[1,] 100.132501 49.831277
[2,]  49.949651 19.893713
[3,]  20.070452  9.899769

    Fixed param inteslope
[1]  2.98483 30.11415

    Knots int
[[1]]
          [,1]
[1,] 5.060082
[2,] 1.990875
[3,] -0.023199

[[2]]
          [,1]
[1,]  50.13250
[2,]  19.796969

    Fixed param int
[1]  9.897868 3.111224

    Knots slope
[[1]]
          [,1]
[1,] 10.026473
[2,] 4.99262
[3,] -2-172738

[[2]]
          [,1]
[1,]  1.002452
```

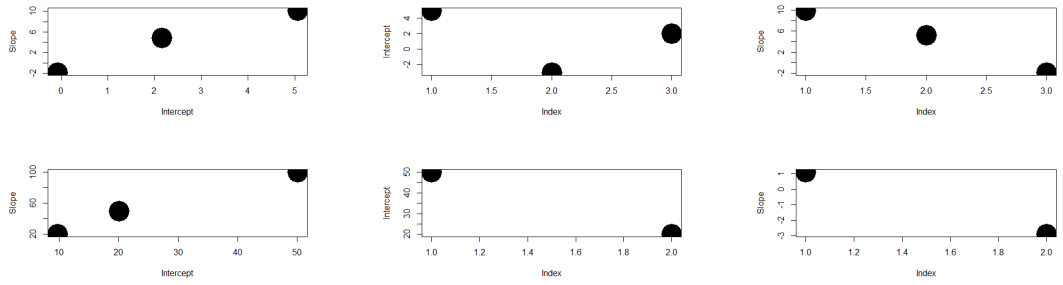
```
[2,]   -2.998364

    Fixed param slope
[1]   0.025362 3.236743
```

Again, it is possible to see the correctness of the result by comparing it with the parameters in the Tables 4.3,4.4 and 4.5.

To display the random parameters estimated by the algorithm we use the function `dotplot.BSPEM` which is invoked via the following code.

```
dotplot.BSPEM(bspmIs)
dotplot.BSPEM(bspmI)
dotplot.BSPEM(bspmS)
```

The result is presented in Fig.4.7. In detail the plot in Fig.4.7a is related to the `inteslope` case, the one in Fig.4.8b to the `int` case and finally the one in Fig.4.8c to the `slope` case.



(a) Dotplot of the knots obtained through the algorithm with the `case` input set to `inteslope`

(b) Dotplot of the knots obtained through the algorithm with the `case` input set to `int`

(c) Dotplot of the knots obtained through the algorithm with the `case` input set to `slope`

Figure 4.7: Dotplot of the random parameters of the model, the width of the dot depends on the related weight of the knot

4.2.5. JMPSPEM example

Finally, we present the case concerning the JMPSEM function. The structure remains unchanged from the previous two. Therefore, we load the data into the workspace and then use it to obtain three JMSPEM objects via the JMSPEM function. The related code is the following:

```
df.inteslope=data(JMSPEM.inteslope.dataset,package="SpMEMs")
```

```
df.int=data(JMSPEM.int.dataset,package="SpMEMs")
df.slope=data(JMSPEM.slope.dataset,package="SpMEMs")

jsmpIs=JMSPEM(case='inteslope',dataset=df,resp_var_name="y",ran_var_name="z
jsmpI=JMSPEM(case='int',dataset=df,resp_var_name="y",groups="groups")
jsmpS=JMSPEM(case='slope',dataset=df,resp_var_name="y",ran_var_name="z",gro
```

Having obtained the three BSPEM objects, it is possible to check the work done by the algorithm by looking at the outputs `knots` and `param`

```
print("Knots␣inteslope")
jspmIs$knots
print("Fixed␣params␣inteslope")
jspmIs$param

print("Knots␣int")
jspmI$knots
print("Fixed␣params␣int")
jspmI$param

print("Knots␣slope")
jspmS$knots
print("Fixed␣params␣␣slope")
jspmS$param
```

we obtain:

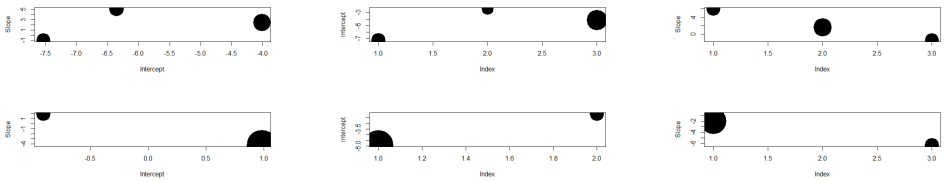
```
    Knots inteslope
[[1]]
           [,1]         [,2]
[1,]  5.102350   -6.320131
[2,]  2.389102   -4.021823
[3,] -0.896743   -7.736252

[[2]]
           [,1]       [,2]
[1,]  1.91526   -0.891627
[2,]  -3.987543   1.072563


    Fixed param inteslope
```

```
[1]   -4.986543 2.086578

    Knots int
[[1]]
           [,1]
[1,] -7.094356
[2,] -4.012635
[3,] -2.956234

[[2]]
           [,1]
[1,]  -4.756789
[2,]  -2.547890

    Fixed param int
[1]  4.098789 -2.876123 2.009432 -2.290912

    Knots slope
[[1]]
           [,1]
[1,] 5.124987
[2,] 2.123647
[3,] -0.876542

[[2]]
           [,1]
[1,]  -2.130599
[2,]  -6.270398

    Fixed param slope
[1]  -1.200498 -3.270368 -2.271070 2.128908
```

The correctness of the result is easily seen by looking at the models in equations 4.13, 4.14 and 4.15. It is possible to display the random parameters through the function dotplot.JMSPEM. In fact via the following command line:

```
dotplot.JMSPEM(jspmIs)
dotplot.JMSPEM(jspmI)
dotplot.JMSPEM(jspmS)
```

we get the plots in Fig.4.8



(a) Dotplot of the knots obtained through the algorithm with the `case` input set to `inteslope`

(b) Dotplot of the knots obtained through the algorithm with the `case` input set to `int`

(c) Dotplot of the knots obtained through the algorithm with the `case` input set to `slope`

Figure 4.8: Dotplot of the random parameters of the model, the width of the dot depends on the related weight of the knot

Finally, it is important to make one last point about the width of the points in the plots in the Fig.4.8. In fact, in this case the weights do not all have the same value, as we can discover with the code below:

```
> jspmIs$weights_marg
[[1]]

[1] 0.279082  0.401053 0.319865

[[2]]

[2] 0.284533 0.715467
```

5 | Conclusion

The R package described in this thesis makes it easier to deal with various case histories involving mixed-effects models. In particular, through to the GMERF and GMET functions, it proposes an alternative approach by estimating the fixed component of the model via tree-object. Applications of the SPEM function and its derivatives, on the other hand, allow the relaxation of the Gaussianity assumptions of the random component of the model giving the possibility of applying mixed-effects models to a larger number of cases. It's crucial to remember that the examples provided are merely a brief introduction to the approaches' possibilities; one more illustration is the use of SPEM techniques with datasets pertaining to Italian education ([16], [17], [18] and [19]). The SPEM and associated methods are distinctive because they are also unsupervised classification techniques, providing these models a dual value. The R SpMEMs package, therefore, represents an important tool for the user that will have the ability to easily access these innovative models. In addition, having presented all the necessary tools for both the theoretical understanding of the methods behind the functions and the understanding of the outputs of the algorithms, users will be facilitated in the analysis by being able to approach it with less difficulty. The next steps involve more attention and explanation of all the various types of errors that can be encountered in using the package and to correcting and improving the functions so that they can also be used in other case studies that received less attention during the thesis work.

Bibliography

[1] A. Agresti. *An introduction to categorical data analysis*. Wiley, 2018.

[2] D. Bates, M. Mächler, B. Bolker, and S. Walker. Fitting linear mixed-effects models using lme4. *Journal of Statistical Software*, 67(1):1–48, 2015. doi: 10.18637/jss.v067.i01.

[3] R. D. Bock and M. Aitkin. Marginal maximum likelihood estimation of item parameters: Application of an em algorithm. *Psychometrika*, 46(4):443–459, 1981.

[4] L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[5] M. Fokkema, N. Smits, A. Zeileis, T. Hothorn, and H. Kelderman. Detecting treatment-subgroup interactions in clustered data with generalized linear mixed-effects model trees. *Behavior research methods*, 50(5):2016–2034, 2018.

[6] L. Fontana, C. Masci, F. Ieva, and A. M. Paganoni. Performing learning analytics via generalised mixed-effects trees. *Data*, 6(7):74, 2021.

[7] L. Fontana, C. Masci, F. Ieva, and A. M. Paganoni. Performing learning analytics via generalised mixed-effects trees. *Data*, 6(7):74, 2021.

[8] A. Hajjem, F. Bellavance, and D. Larocque. Mixed effects regression trees for clustered data. *Statistics & probability letters*, 81(4):451–459, 2011.

[9] A. Hajjem, F. Bellavance, and D. Larocque. Mixed-effects random forest for clustered data. *Journal of Statistical Computation and Simulation*, 84(6):1313–1328, 2014.

[10] A. Hajjem, D. Larocque, and F. Bellavance. Generalized mixed effects regression trees. *Statistics & Probability Letters*, 126:114–118, 2017.

[11] A. Liaw and M. Wiener. *randomForest: Breiman and Cutler's Random Forests for Classification and Regression*, 2021. URL `https://cran.r-project.org/web/packages/randomForest/index.html`.

[12] L. I. Lin et al. Total deviation index for measuring individual agreement with appli-

cations in laboratory performance and bioequivalence. *Statistics in Medicine*, 19(2): 255–270, 2000.

[13] B. G. Lindsay et al. The geometry of mixture likelihoods: a general theory. *The annals of statistics*, 11(1):86–94, 1983.

[14] B. G. Lindsay et al. The geometry of mixture likelihoods, part ii: the exponential family. *The Annals of Statistics*, 11(3):783–792, 1983.

[15] W.-Y. Loh. Classification and regression trees. *Wiley interdisciplinary reviews: data mining and knowledge discovery*, 1(1):14–23, 2011.

[16] C. Masci, A. M. Paganoni, and F. Ieva. Semiparametric mixed effects models for unsupervised classification of Italian schools. *Journal of the Royal Statistical Society Series A*, 182(4):1313–1342, October 2019. doi: 10.1111/rssa.12449. URL `https://ideas.repec.org/a/bla/jorssa/v182y2019i4p1313-1342.html`.

[17] C. Masci, F. Ieva, T. Agasisti, and A. M. Paganoni. Evaluating class and school effects on the joint student achievements in different subjects: a bivariate semiparametric model with random coefficients. *Computational Statistics*, 36(4):2337–2377, Dec 2021. ISSN 1613-9658.

[18] C. Masci, F. Ieva, and A. M. Paganoni. Semiparametric multinomial mixed-effects models: A university students profiling tool. *The Annals of Applied Statistics*, 16(3): 1608 – 1632, 2022. doi: 10.1214/21-AOAS1559. URL `https://doi.org/10.1214/21-AOAS1559`.

[19] F. P. A. Masci, C. Ieva. A multinomial mixed-effects model with discrete random effects for modelling dependence across response categories. 2022.

[20] C. McCulloch, H. Lin, E. Slate, and B. Turnbull. Discovering subpopulation structure with latent class mixed models. *Statistics in medicine*, 21(3):417–429, 2002.

[21] B. Muthén. Latent variable analysis. *The Sage handbook of quantitative methodology for the social sciences*, 345:368, 2004.

[22] B. Muthén and K. Shedden. Finite mixture modeling with mixture outcomes using the em algorithm. *Biometrics*, 55(2):463–469, 1999.

[23] D. S. Nagin. Analyzing developmental trajectories: a semiparametric, group-based approach. *Psychological methods*, 4(2):139, 1999.

[24] M. Pellagatti, C. Masci, F. Ieva, and A. M. Paganoni. Generalized mixed-effects random forest: A flexible approach to predict university student dropout. *Stat. Anal.*

Data Min., 14(3):241–257, may 2021. ISSN 1932-1864. doi: 10.1002/sam.11505. URL https://doi.org/10.1002/sam.11505.

[25] J. C. Pinheiro and D. M. Bates. Linear mixed-effects models: basic concepts and examples. *Mixed-effects models in S and S-Plus*, pages 3–56, 2000.

[26] C. Proust-Lima, L. Letenneur, and H. Jacqmin-Gadda. A nonlinear latent class model for joint analysis of multivariate longitudinal data and a binary outcome. *Statistics in Medicine*, 26(10):2229–2245, 2007.

[27] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2014. URL http://www.R-project.org/.

[28] R Core Team. *stats: The R Stats Package*, 2022. URL https://stat.ethz.ch/R-manual/R-devel/library/stats/html/00Index.html. R package version 4.1.2.

[29] B. Ripley and B. Venables. Modern applied statistics with s. *Springer New York*, 2002. URL https://www.stats.ox.ac.uk/pub/MASS4/.

[30] R. J. Sela and J. S. Simonoff. Re-em trees: a data mining approach for longitudinal and clustered data. *Machine learning*, 86(2):169–207, 2012.

[31] R. J. Sela and J. S. Simonoff. Re-em trees: a data mining approach for longitudinal and clustered data. *Machine learning*, 86(2):169–207, 2012.

[32] J. L. Speiser, B. J. Wolf, D. Chung, C. J. Karvellas, D. G. Koch, and V. L. Durkalski. Bimm tree: a decision tree method for modeling clustered and longitudinal binary outcomes. *Communications in Statistics-Simulation and Computation*, pages 1–20, 2018.

[33] T. Therneau and B. Atkinson. *Recursive Partitioning and Regression Trees*, 2021. URL https://cran.r-project.org/web/packages/rpart/index.html. R package version 4.1-15.

[34] J. K. Vermunt and J. Magidson. Latent class cluster analysis. *Applied latent class analysis*, 11:89–106, 2002.

Acknowledgements

I want to express my gratitude to my thesis advisor, Professor Chiara Masci, and co-advisor, Professor Francesca Ieva, for their assistance. This period was calm and tranquil because of their complete availability, which is unusual when it comes to degree theses. I want to thank Professor Masci once again for allowing me to see academic work in a new light because of her passion.