The mcemGLM package

Felipe Acosta Archila

August 13, 2015

1 A Generalized Linear Mixed Model

We start by assuming that we observe a vector of data $Y = (Y_1, \dots, Y_n)$ corresponding to a probability model that depends on a $(p + l)$-dimensional parameter vector θ, a known $n \times p$ fixed effects design matrix X, a known $n \times k$ known random effects design matrix Z , and a k-dimensional vector of unobservable random effects U.

Let θ consist of p fixed effects coefficients $\beta = (\beta_1, \dots, \beta_p)^T$ and l variance parameters, $\sigma^2 = (\sigma_1^2, \dots, \sigma_l^2)^T$, associated to the random effects U. Our goal is to find maximum likelihood estimates (MLEs) for the $(p + l)$-dimensional parameter θ in a space Θ for a generalized linear mixed model.

We assume that the expected variable of Y_i, can be written as a linear combination of the observable and unobservable variables through a bijective “link” function g. Let X_i and Z_i be the ith rows of the matrices X and Z, and let $\mathrm{E}(Y_i|U = u) = \mu_i$. Then

$$g(\mu_i) = X_i\,\beta + Z_i\,u, \text{ for } i = 1, \dots, n.$$

Let $U = (U_1^T, \dots, U_l^T)^T$, and $Z = (Z_1 \cdots Z_l)$ a decomposition for the vector U and the matrix Z. We assume that U_i is a k_i-dimensional vector with $\sum_i^l k_i = k$. Furthermore we assume that U_i has a known distribution with variance that depends on the parameter σ_i^2. In general let $\mu = (\mu_1, \dots, \mu_n)$ and let $g(\mu)$ denote the element-wise evaluation of g on the vector μ, then we can write mean our model as

$$g(\mu) = X\,\beta + \sum_{i=1}^{l} Z_i\, u_i. \tag{1}$$

Let $h_U(u)$ be the probability density function of U. We assume that conditional on U, the data is generated from a probability model with probability mass function $f(Y|\theta, X, Z, U)$ and that we can write its likelihood function in terms of $\mu = g^{-1}(X\,\beta + \sum_{i=1}^{l} Z_i\, u_i)$, and σ^2. Defining the model this way yields to the following likelihood functions:

1. A complete likelihood function:

$$L(\theta|X, Z, U) = f(y, u|\theta, X, Z) = f_Y(y|\theta, X, Z, u)\, h_U(u|\theta). \tag{2}$$

2. And a marginal likelihood function:

$$L(\theta|X, Z) = \int_{\mathbb{R}^l} f(Y|\theta, X, Z, U)\, h_U(u|\theta) du. \tag{3}$$

Since the vector U is not observable we need to obtain the MLEs from 3. This means that before maximizing the likelihood function we need to integrate out the vector of random effects.

The `mcemGLM` package focuses on three types of models for the marginal data:

1. Bernoulli data. We say that $Y_i \overset{iid}{\sim}$ Bernoulli(p_i), for $i = 1, \ldots, n$, with $0 < p_i < 1$, if Y_i has probability mass function

$$f(y_i) = p_i^{y_i}(1 - p_i)^{1-y_i}, \text{ for } y_i = 0, 1.$$

With $\mathrm{E}(Y_i) = p_i$, $\mathrm{Var}(Y_i) = p_i(1 - p_i)$, and $g(p_i) = \log(p_i/(1 - p_i))$.

2. Poisson data. We say that $Y_i \overset{iid}{\sim}$ Poisson(μ_i) for $i = 1, \ldots, n$, if Y_i has probability mass function

$$f(y_i) = e^{-\mu_i} \frac{\mu_i^{y_i}}{y_i!}, \text{ for } y_i = 0, 1, 2, \ldots$$

With $\mathrm{E}(Y_i) = \mu_i$, $\mathrm{Var}(Y_i) = \mu_i$, and $g(\mu_i) = \log(\mu_i)$.

3. Negative binomial data. We say that $Y_i \overset{iid}{\sim}$ neg-binom(μ_i, α), for $i = 1, \dots, n$, with $\mu_i > 0$, and $\alpha > 0$, if Y_i has probability mass function

$$f(y_i) = \frac{\Gamma(y_i + \alpha)}{\Gamma(\alpha)\, y_i!} \left(\frac{\alpha}{\mu_i + \alpha}\right)^{\alpha} \left(\frac{\mu_i}{\mu_i + \alpha}\right)^{y_i}, \text{ for } y_i = 0, 1, 2, \dots$$

With $\mathrm{E}(Y_i) = \mu_i$, $\mathrm{Var}(Y_i) = \mu_i + \mu_i^2/\alpha$, and $g(\mu_i) = \log(\mu_i)$.

The expectation and variance of Y_i can be found easily by using iterated expectation with respect to a random variable M distributed gamma with shape parameter α, and rate parameter α/μ and setting $Y_i|M = m \sim$ Poisson(m).

By using this definition of the distribution of Y_i we can treat the parameter α as the amount of over-dispersion with respect to the Poisson distribution. The value $\alpha = \infty$ corresponds to no over-dispersion.

By introducing α to the model notice that we need to estimate this extra parameter in addition to β and σ^2.

In addition to the model selection the `mcemGLM` package allows to specify two types of random effects. Let I_k be an $n \times n$ identity matrix, $\mathrm{N}_n(\mu, \Sigma)$ an n-dimensional multivariate normal distribution with mean vector μ and covariance matrix Σ, and $t_n(\nu, \mu, \Sigma)$ an n-dimensional multivariate t distribution with ν degrees of freedom, location vector μ and scale matrix Σ.

1. Normal distribution. We set $U_i \sim \mathrm{N}_{k_i}(0, \sigma_i^2 I_{k_i})$ for $i = 1, \dots, l$, and set the U_is to be jointly independent.

2. t distribution with known degrees of freedom ν. We set $U_i \sim t_{k_i}(\nu, 0, \sigma_i^2 I_{k_i})$ for $i = 1, \dots, l$, and set the U_is to be jointly independent.

2 The MCEM algorithm

The MCEM algorithm is a modification of the EM algorithm. The later assumes two sets of data an observed data set Y and a set of missing data U.

The EM algorithm estimates the MLEs by an iterative algorithm. Let $\theta^{(t)}$ denote the current estimate at the ith iteration. Let

$$Q(\theta, \theta^{(t)}) = \mathrm{E}\left[\log f(y, u|\theta, X, U)|y, \theta^{(t)}\right]. \tag{4}$$

The next value, $\theta^{(t+1)}$, is found by maximizing 4 with respect to θ. The expectation in 4 is taken with respect to $f(u|y, \theta, X, Z)$ hence if we want to obtain its closed form we need $f(y, u|\theta, X, Z)$ and $f_Y(y|\theta, X, Z)$. The later function is not available for the models we are considering, therefore we need to resort to a numerical method to calculate this expectation.

The solution implemented in the `mcemGLM` package is to estimate 4 via a Markov chain Monte Carlo (MCMC) step. This works by obtaining a sample $u_{t,1}, \dots, u_{t,m}$ from a Markov chain with stationary distribution $f(u|y, \theta, X, Z)$ and then maximizing

$$\widehat{Q}(\theta) = \sum_{j=1}^{m} \log f(y, u_{t,j}|\theta, X, Z) \tag{5}$$

with respect to θ to obtain $\theta^{(t+1)}$.

The algorithm is run until a termination condition has been reached or the maximum number of iterations has been done.

3 The mcemGLM package

The package runs through the following steps:

1. Choose a starting value. The default method is to fit a model without random effects and using the MLEs of the fixed coefficients as starting values for β. For σ we set a predefined value of 5.

2. Obtain the sample $u_{t,1}, \dots, u_{t,m}$. This is done by using a Metropolis–Hastings algorithm that uses a multivariate normal random variable as its proposal. The standard deviation vector of the proposal distribution is chosen by performing an auto–tuning step before the first iteration. After each iteration the rejection rate of the chain is checked and if it is either too large (> 0.4) or to small (< 0.1) the package performs an auto-tuning step before the next iteration.

3. After obtaining the sample 5 is maximized with respect all the parameters using the `trust` function from the `trust` package. The maximizers are set as the current value of the estimator

of the MLEs.

4. Steps 2 and 3 are repeated until the condition

$$\max_i \left\{ \frac{|\theta_i^{(t)} - \theta_i^{(t-1)}|}{|\theta_i^{(t)}| + \delta} \right\} < \epsilon$$

for specified values of δ and ϵ is met three consecutive times or a maximum number of iterations have been performed.

The default values in the package are $\delta = 0.05$ and $\epsilon = 0.01$ but these can be easily changed by the user. The default number of iterations is 80 and this value can also be changed by the user.

5. After terminating the iterative process another sample from the random effects is obtained to estimate the information matrix of the model.

4 Using the mcemGLM package

```
> require(mcemGLM)
> data("simData.rdata")
> head(simData)
  obs        x1       x2     x3 z1 z2 z3 count count2
1   0  9.571463 5.924451    red D1  1  A     5      5
2   0 10.062451 5.358087 yellow D1  1  A     4     24
3   0  8.020461 4.755584 yellow D1  1  A     4      4
4   0 10.842312 5.610179 yellow D1  1  A     3      9
5   0  8.457872 2.882771 yellow D1  1  A     3      6
6   0 11.154501 4.942196 yellow D1  1  A     3      6
> summary(simData)
      obs             x1                x2              x3        z1      z2
 Min.   :0.00   Min.   : 7.338   Min.   :1.784   blue  :65   D1:40   1:50
 1st Qu.:0.00   1st Qu.: 9.332   1st Qu.:4.405   red   :67   D2:40   2:50
 Median :0.00   Median : 9.995   Median :5.119   yellow:68   D3:40   3:50
 Mean   :0.39   Mean   : 9.990   Mean   :5.148               D4:40   4:50
 3rd Qu.:1.00   3rd Qu.:10.629   3rd Qu.:5.808               D5:40
 Max.   :1.00   Max.   :13.533   Max.   :7.531
 z3           count           count2
 A:100   Min.   : 0.0   Min.   :  0.00
 B: 60   1st Qu.: 3.0   1st Qu.:  6.00
 D: 40   Median : 4.5   Median : 12.00
         Mean   : 5.1   Mean   : 16.16
         3rd Qu.: 7.0   3rd Qu.: 20.00
         Max.   :18.0   Max.   :108.00
```

The data consist of three fixed effects, $x1$, $x2$, and $x3$. The first two fixed effects are continuous and $x3$ is a factor with three levels. There are three variables that we can use as variance components $z1$ (5 levels), $z2$ (4 levels), and $z3$ (3 levels.) The component $z2$ can be nested within $z1$ and $z3$ is crossed with these.

First we will consider a simple model based on this data using `obs` as the binary response.

4.1 A simple model

We will fit a model with one variance component, $z3$ and we will consider $z1$ as a fixed effect along with $x1$.

The main model arguments for the `mcemGLMM` function are `fixed` and `random`. These specify the fixed and random effects of the model. The response must be included in the `fixed` argument. In this first example we are considering $x1$ and $z1$ as fixed and $z3$ as random. We can fit this model with the following command:

```
> fit0 <- mcemGLMM(fixed = obs ~ x1 + z1,
+                 random = ~0+z2,
+                   data = simData,
+                 family = "bernoulli",
+                 vcDist = "normal")
```

The rest of the used arguments are:

- `data:` argument contains the name of the data frame with the data.
- `family:` argument specifies the type of model to be fitted. The options are "bernoulli" for logistic regression, "poisson" for Poisson count regression, and "negbinom" for negative binomial count regression.
- `vcDist:` argument specifies the distribution of the random effects. The option are "normal", and "t". In case of t random effects an extra argument with the degrees of freedom must be supplied.

We can start by taking a look at the coefficient and variance estimates with the `summary` command:

```
> summary(fit0)

Call:
  mcemGLMM(fixed = obs ~ x1 + z1, random = ~0 + z2, data = simData,
    family = "bernoulli", vcDist = "normal")

   Two sided Wald tests for fixed effects coefficients:

              Estimate Std. Error    z value   Pr(>|z|)
(Intercept)  2.3683592  1.6681565  1.4197464 0.15568152
x1          -0.3368267  0.1640034 -2.0537785 0.03999714
z1D2         1.3871167  0.4908233  2.8261021 0.00471182
z1D3        -1.0466774  0.6047379 -1.7307951 0.08348832
z1D4         0.4811148  0.4944886  0.9729543 0.33057603
z1D5         1.3291029  0.4907177  2.7084876 0.00675906



   One sided Wald tests for variance components:

    Estimate Std. Error   z value    Pr(>z)
z2 0.1080782  0.1508095 0.7166536 0.2367939
```

We first get a print of the original call used to fit the model. The summary print out has two tables. The first table shows the estimates, standard errors and z tests for the fixed effect coefficients. While the second table contains the same information but for the variance estimates.

Now we can look at an ANOVA table based on Wald tests.

```
> anova(fit0)
```

```
   Wald's Chi-squared ANOVA table


   Df Wald Stat     Pr(>W)
x1  1  4.218006 0.03999714
z1  4 24.562532 0.00006160
```

Each line corresponds to a test on the coefficients that are related to each variable. In the case of a continuous variable or a binary this is equivalent to the z test performed with `summary`. When a categorical variable has more than two categories `anova` will test run a chi–squared test on all the coefficients that are related to that variable. In this case the chi–squared test for $z1$ tests if the corresponding coefficients for `D2`, D3 , `D4`, and `D5` are both equal to zero.

We can run multiple comparison tests for the levels of $z1$. First we need to create a contrast matrix with each row representing a contrast that we want to test. In this case

```
> ctr0 <- rbind("D1 - D2" = c(0, 0,-1, 0, 0, 0),
+                "D1 - D3" = c(0, 0, 0,-1, 0, 0),
+                "D1 - D4" = c(0, 0, 0, 0,-1, 0),
+                "D1 - D5" = c(0, 0, 0, 0, 0,-1),
+                "D2 - D3" = c(0, 0, 1,-1, 0, 0),
+                "D2 - D4" = c(0, 0, 1, 0,-1, 0),
+                "D2 - D5" = c(0, 0, 1, 0, 0,-1),
+                "D3 - D4" = c(0, 0, 0, 1,-1, 0),
+                "D3 - D5" = c(0, 0, 0, 1, 0,-1),
+                "D4 - D5" = c(0, 0, 0, 0, 1,-1))
```

Notice that rows one and two are the contrasts that that compare the baseline, `D1`, to the other levels, hence these will have equivalent test statistics as those obtained in `summary`. However `contrasts.mcemGLMM` accounts for multiple comparisons by adjusting the p–values via Bonferroni correction therefore it is possible to obtain significance in `summary` and not in `contrasts.mcemGLMM` since this p–value will likely be larger.

```
> contrasts.mcemGLMM(object = fit0, ctr.mat = ctr0)

           Estimate Std. Err.        Wald Adj. p-value
D1 - D2 -1.38711666 0.4908233  7.98685283 0.0471182493
D1 - D3  1.04667741 0.6047379  2.99565159 0.8348832049
D1 - D4 -0.48111479 0.4944886  0.94664003 1.0000000000
D1 - D5 -1.32910289 0.4907177  7.33590494 0.0675906430
D2 - D3  2.43379408 0.5883939 17.10926441 0.0003528974
D2 - D4  0.90600187 0.4715595  3.69134908 0.5469535980
D2 - D5  0.05801377 0.4654850  0.01553284 1.0000000000
D3 - D4 -1.52779221 0.5909675  6.68346045 0.0973115299
D3 - D5 -2.37578031 0.5894285 16.24614167 0.0005562275
D4 - D5 -0.84798810 0.4707865  3.24437836 0.7166887291
```

For this simple model it is possible to plot the predicted probabilities for each level of `z1`. These estimates correspond to the population means, i.e., the random effects have been set to zero. Figure 4.1 shows the plots of the fitted probabilities as a function of $x1$ for the different levels of $z1$.

We can calculate the Pearson and deviance residuals of the model with the `residuals` command. Figures 4.1 and 4.1 shows these plots.

To assess convergence we can look at trace plots of the MLE estimates and the value of the loglikelihood function across the EM iterations. These are stored in the `mcemGLMM` object returned by the function the `mcemGLMM` function on the fields `mcemEST` and `loglikeVal`. Figure 4.1 shows trace plots at each EM iteration of these quantities.

We can also take a look at the trace plots of the Markov chain used to estimate the Q function. Since this approximates an integral of dimension equal to the number of random effects it might not be practical to look at all the chains. The last MCMC step is saved on the field `randeff` as a matrix. Each column of this matrix corresponds to one random effect.

This matrix can be used to get predictions of the observed random effects.

```
> plot(simData$x1, predict(fit0, type = "response"), col = simData$z1, xlab = "x1")
```

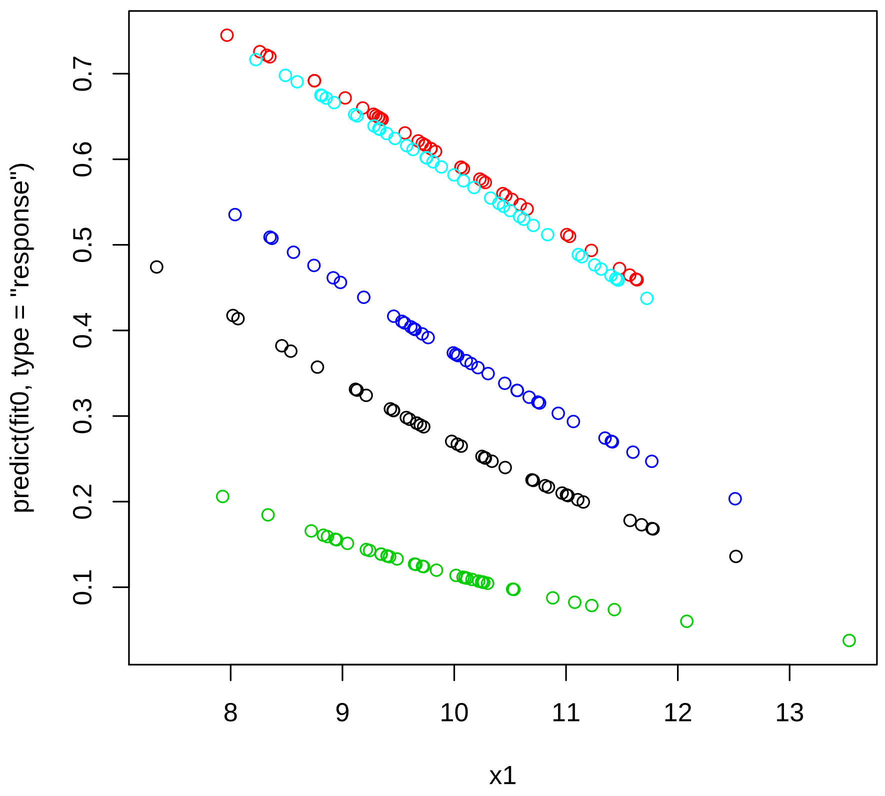


Figure 1: Fitted probabilities. Each color represent a level of $z1$

```
> colMeans(fit0$randeff)

       z21        z22        z23        z24
-0.0427675 -0.1262861  0.3917916 -0.2262055
```

To see the sampling on the loglikelihood function we can plot the values of the complete loglikelihood at each MCMC step of the last EM iteration. These values are stored in the `loglikeMCMC` field of the `mcemGLMM` object.

```
> plot(simData$x1, residuals(fit0, type = "deviance"))
```

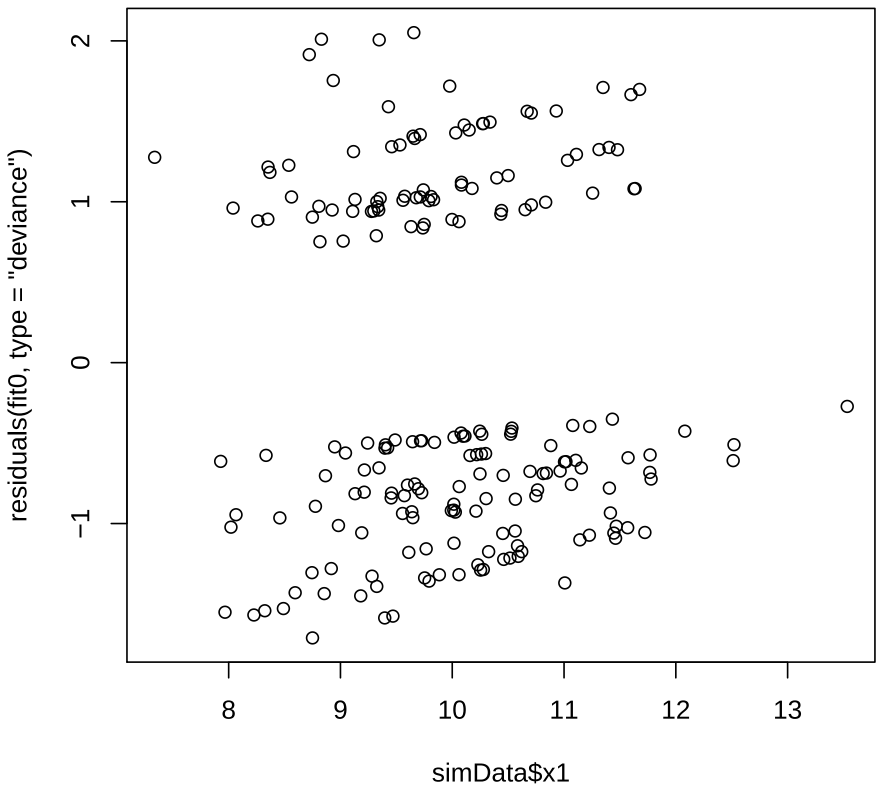


Figure 2: Deviance residuals

```
> plot(simData$x1, residuals(fit0, type = "pearson"))
```

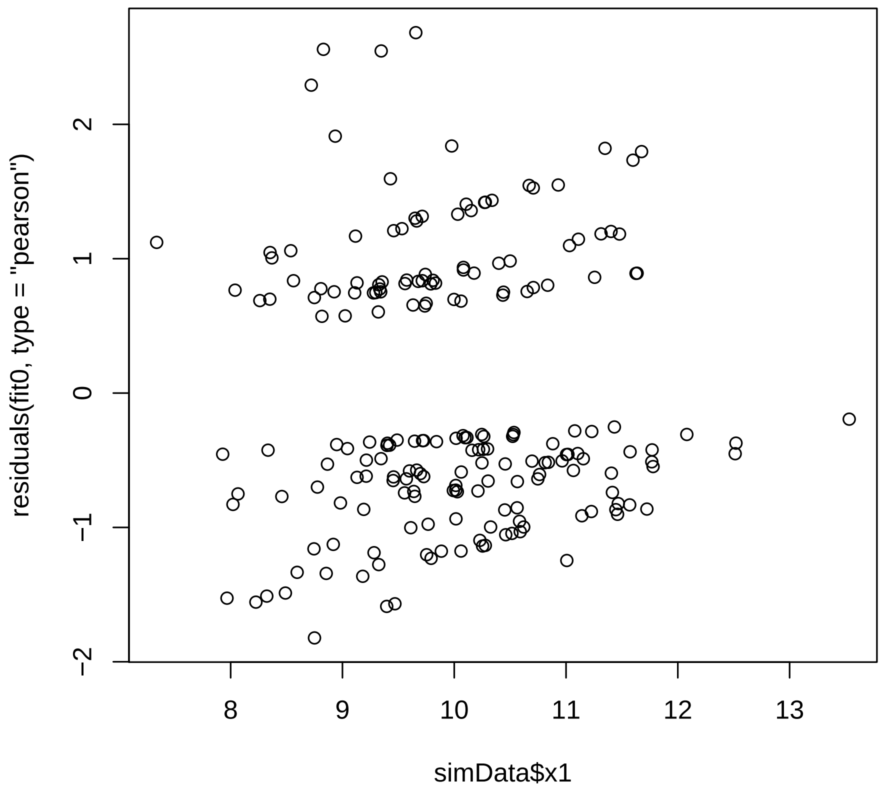


Figure 3: Pearson residuals

```
> par(mfrow = c(1, 2))
> ts.plot(fit0$mcemEST, main = "MLEs estimates",
+         xlab = "EM Iteration", ylab = "MLE value")
> ts.plot(fit0$loglikeVal, main = "Loglikelihood values",
+         xlab = "EM iteration", ylab = "Likelihood")
```

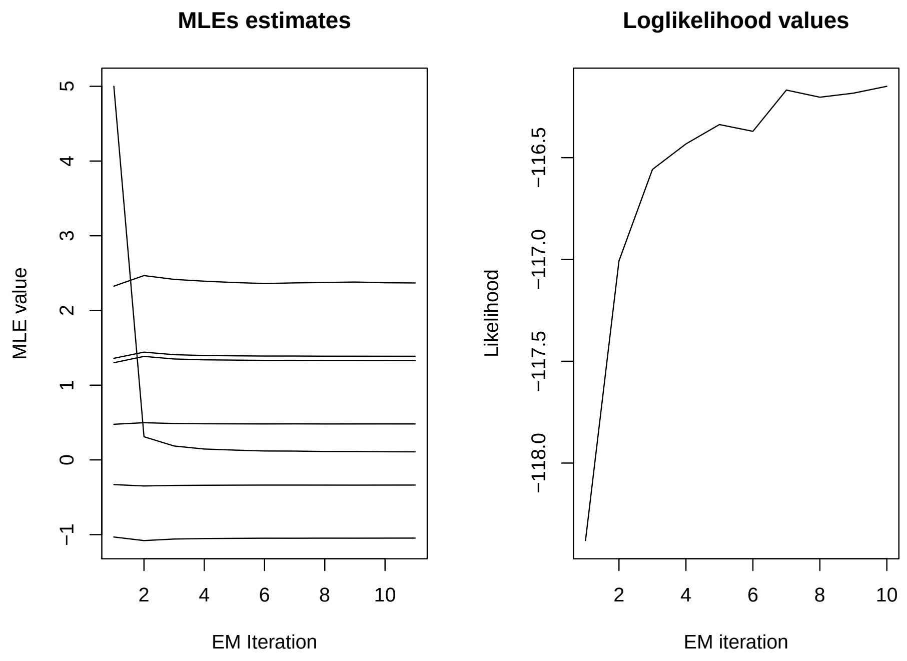


Figure 4: MLEs estimates (left) and Loglikelihood value (right) after each EM iteration.

```
> ts.plot(fit0$randeff[, 1], xlab = "MCMC iteration")
```

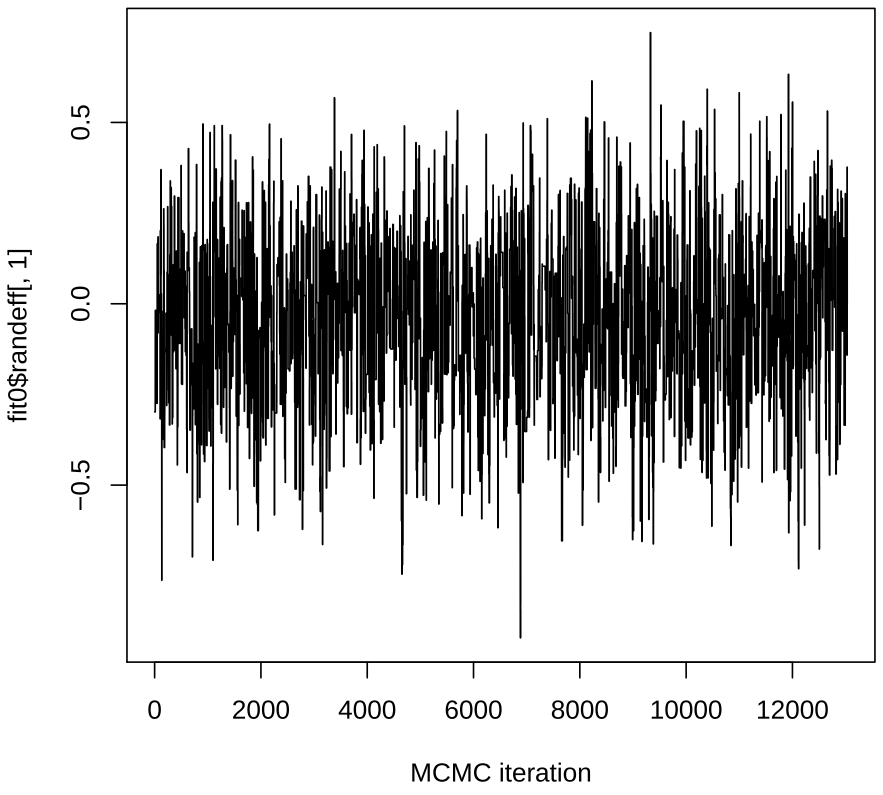


Figure 5: Trace plot for MCMC output for the first random effect.

```
> ts.plot(fit0$loglikeMCMC, xlab = "MCMC iteration")
```

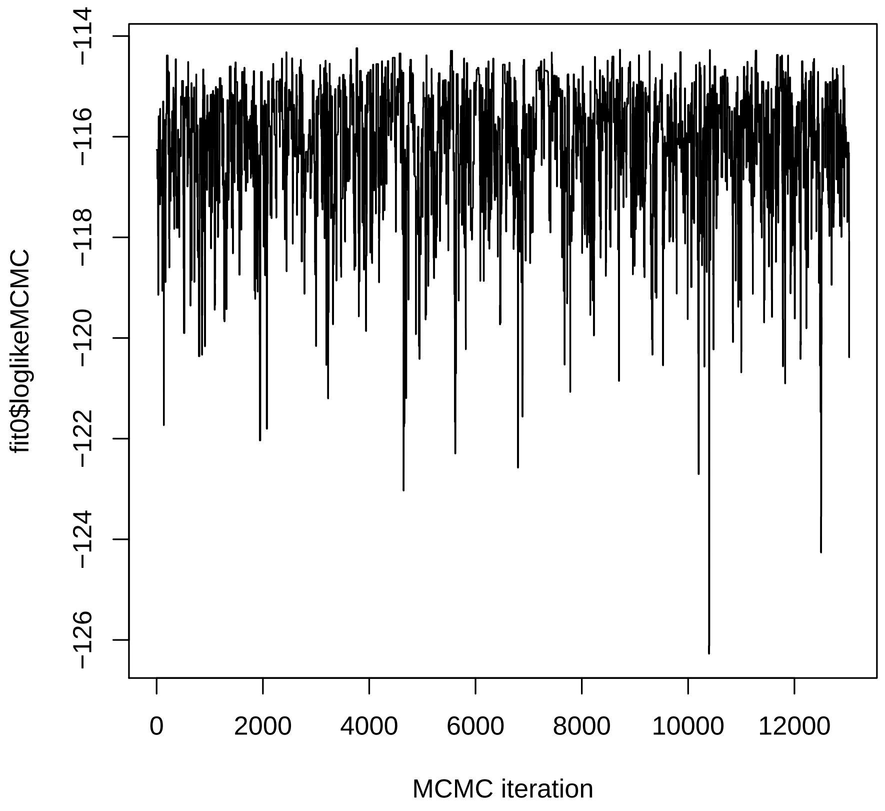


Figure 6: Trace plot for the complete loglikelihood function.

4.2 Fitting a more complex model

To specify more than one random effect we need to put them into a list and state that there is no intercept for that effect. In case of nested random effects if labels are repeated across it is necessary to fit the lower level by using the interaction with the upper level.

In this specific example, the labels for $z2$, "1", "2", "3", and "4", are used for each level of $z1$. If the labels of $z2$ are unique within $z1$ it is not necessary to use the interaction term. However it is recommended to use the interaction form for the sake of clarity in the model statement.

```
> fit1 <- mcemGLMM(fixed = obs ~ x1 + x2 + x3,
+                 random = list(~0+z1, ~0+z1:z2),
+                   data = simData,
+                 family = "bernoulli",
+                 vcDist = "t",
+                 df = c(5, 5))
```

The `df` argument specifies the degrees of freedom for each variance component in `random.` If `vcDist` is "normal" this argument is not needed.

We can look at the summary and ANOVA of the model as before

```
> summary(fit1)

Call:
  mcemGLMM(fixed = obs ~ x1 + x2 + x3, random = list(~0 + z1, ~0 +
    z1:z2), data = simData, family = "bernoulli", vcDist = "t",
    df = c(5, 5))


   Two sided Wald tests for fixed effects coefficients:


              Estimate Std. Error    z value   Pr(>|z|)
(Intercept)  1.0482656  2.1354187  0.4908946 0.62350096
x1          -0.5100424  0.1936848 -2.6333633 0.00845439
x2           0.5790644  0.1984313  2.9182112 0.00352046
```

```
x3red        0.2188118  0.4477685  0.4886716 0.62507423
x3yellow     0.8289974  0.4815292  1.7215932 0.08514324


   One sided Wald tests for variance components:

       Estimate Std. Error   z value    Pr(>z)
z1    0.5924914  0.9113657 0.6501138 0.2578093
z1:z2 1.3353322  1.1893710 1.1227213 0.1307779
> anova(fit1)
   Wald's Chi-squared ANOVA table

   Df Wald Stat     Pr(>W)
x1  1  6.934602 0.00845439
x2  1  8.515957 0.00352046
x3  2  3.139481 0.20809921
```

We can run multiple comparison tests for the levels of $x3$ as before

```
> ctr1 <- rbind(   "blue - red" = c(0, 0, 0,-1, 0),
+               "blue - yellow" = c(0, 0, 0, 0,-1),
+                "red - yellow" = c(0, 0, 0, 1,-1))
> contrasts.mcemGLMM(object = fit1, ctr.mat = ctr1)

                Estimate Std. Err.      Wald Adj. p-value
blue - red    -0.2188118 0.4477685 0.2387999    1.0000000
blue - yellow -0.8289974 0.4815292 2.9638830    0.2554297
red - yellow  -0.6101857 0.4677397 1.7018277    0.5761474
```

Instead of performing a Wald test to test a fixed effect it is possible to run a likelihood ratio test between two nested models. First we will fit a model without `x3`:

```
> fit2 <- mcemGLMM(fixed = obs ~ x1 + x2,
+                  random = list(~0+z1, ~0+z1:z2),
+                    data = simData,
+                  family = "bernoulli",
+                  vcDist = "t",
+                      df = c(5, 5),
+                  controlEM = list(EMit = 3))
```

Now we can use the `anova` command to run the likelihood ratio test

```
> anova(fit1, fit2)

  Test statistic value: 6.61765024168028
  Degrees of freedom: 2
  p value: 0.0365591
```

4.3 A Poisson model

To fit a Poisson model we only need to change the `family` argument in the `mcemGLMM` command. As an example we will use the `count` variable in `simData`.

```
> fit3 <- mcemGLMM(fixed = count ~ x1 + x2 + x3,
+                  random = list(~0+z2),
+                    data = simData,
+                  family = "poisson",
+                  vcDist = "normal")
```

All the previous methods are available for this type of model.

```
> summary(fit3)

Call:
  mcemGLMM(fixed = count ~ x1 + x2 + x3, random = list(~0 + z2),
    data = simData, family = "poisson", vcDist = "normal")
```

```
   Two sided Wald tests for fixed effects coefficients:

                Estimate Std. Error     z value   Pr(>|z|)
(Intercept)  2.429668527 0.35016284  6.93868171 0.00000000
x1          -0.100594578 0.03139176 -3.20449005 0.00135302
x2           0.042484752 0.03066740  1.38533934 0.16594873
x3red       -0.072106883 0.07945201 -0.90755269 0.36411460
x3yellow     0.002406589 0.07747179  0.03106407 0.97521845


   One sided Wald tests for variance components:

      Estimate  Std. Error   z value    Pr(>z)
z2 0.003188614 0.006782275 0.4701393 0.3191277
```

```
> anova(fit3)

   Wald's Chi-squared ANOVA table

   Df Wald Stat     Pr(>W)
x1  1 10.268757 0.00135302
x2  1  1.919165 0.16594873
x3  2  1.162212 0.55927940
```

```
> contrasts.mcemGLMM(object = fit3, ctr.mat = ctr1)

                  Estimate  Std. Err.         Wald Adj. p-value
blue - red     0.072106883 0.07945201 0.8236518816            1
blue - yellow -0.002406589 0.07747179 0.0009649761            1
red - yellow  -0.074513472 0.07727556 0.9297908976            1
```

```
> plot(simData$x1, predict(fit3),
+      main = "Predicted response values", xlab = "x1")
```

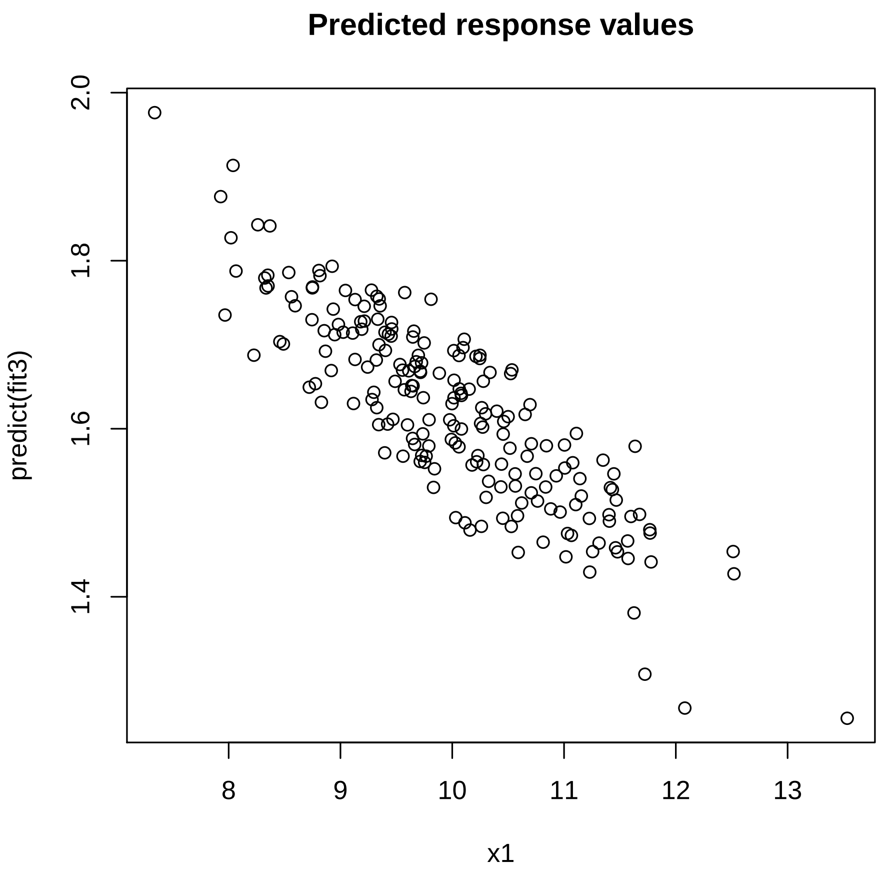


Figure 7: Predicted values as a function of $x1$.

4.4 A negative binomial model

To fit a negative binomial model we need to specify the `family` argument to `"negbinom"`. All the previous methods ara available for this model. When we look at the summary of this model we get an estimate of the overdispersion parameter and its standard error.

```
> fit4 <- mcemGLMM(fixed = count2 ~ x1 + x2 + x3,
+                  random = list(~0+z1, ~0+z1:z2),
+                    data = simData,
+                  family = "negbinom",
> par(mfrow = c(1, 2))
> ts.plot(fit3$mcemEST, main = "MLEs estimates",
+         xlab = "EM Iteration", ylab = "MLE value")
> ts.plot(fit3$loglikeVal, main = "Loglikelihood values",
+         xlab = "EM iteration", ylab = "Likelihood")
```

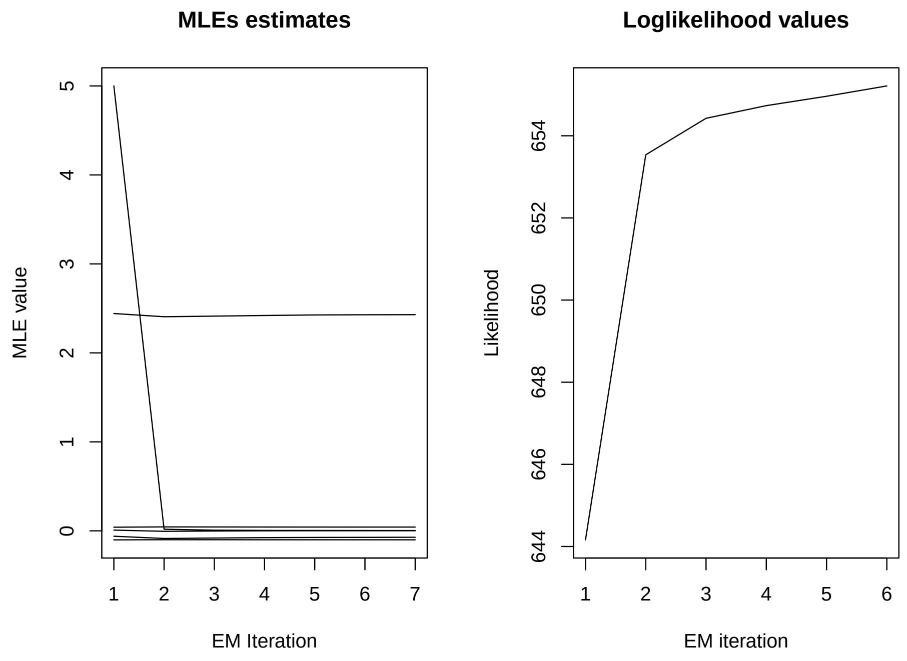


Figure 8: MLEs estimates (left) and Loglikelihood value (right) after each EM iteration.

```
+                 vcDist = "normal")

> summary(fit4)

Call:

  mcemGLMM(fixed = count2 ~ x1 + x2 + x3, random = list(~0 + z1,

    ~0 + z1:z2), data = simData, family = "negbinom", vcDist = "normal")


   Two sided Wald tests for fixed effects coefficients:


               Estimate Std. Error     z value   Pr(>|z|)

(Intercept)  4.21614618 0.57654418  7.31278944 0.00000000

x1          -0.15975518 0.04954114 -3.22469750 0.00126106
```

```
x2           0.01458830 0.04912726  0.29694920 0.76650529

x3red       -0.13490761 0.12613565 -1.06954383 0.28482469

x3yellow    -0.01146224 0.13047303 -0.08785144 0.92999476


   Overdispersion parameter beta:


      Estimate Std. Error

theta 1.388511  0.1252373




   One sided Wald tests for variance components:


        Estimate Std. Error   z value     Pr(>z)

z1    0.05705629 0.09288859 0.6142443 0.26952695

z1:z2 0.18417139 0.08842985 2.0826835 0.01864004
```

```
> anova(fit4)
```

```
   Wald's Chi-squared ANOVA table


   Df   Wald Stat     Pr(>W)

x1  1 10.39867396 0.00126106

x2  1  0.08817883 0.76650529

x3  2  1.39625335 0.49751644
```

```
> contrasts.mcemGLMM(object = fit4, ctr.mat = ctr1)
```

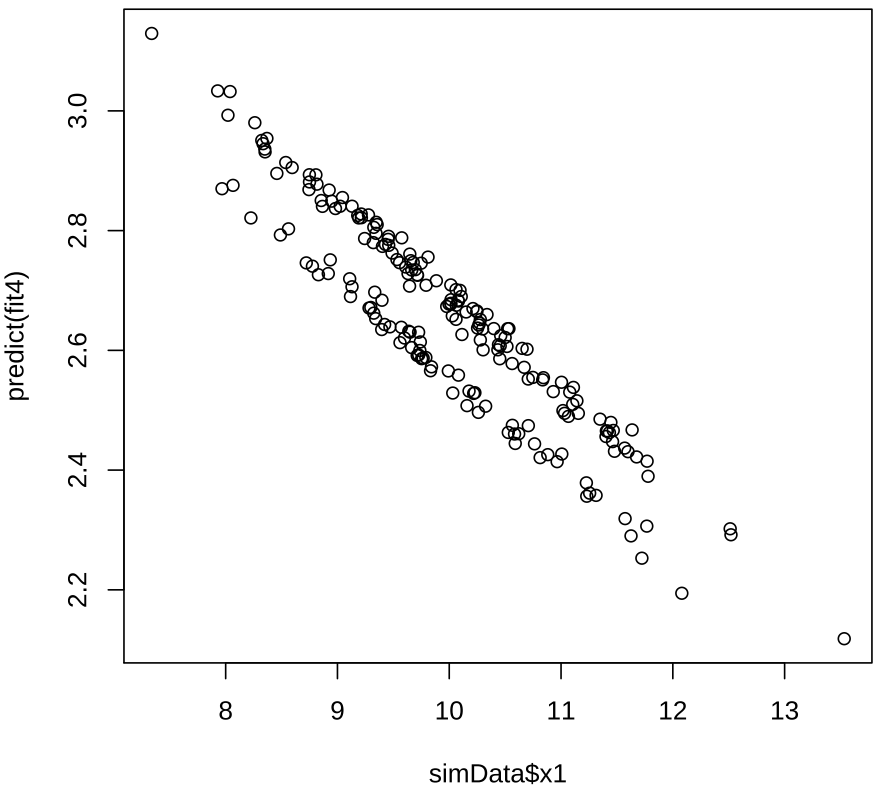
```
                 Estimate Std. Err.        Wald Adj. p-value

blue - red     0.13490761 0.1261357 1.143924013    0.8544741

blue - yellow  0.01146224 0.1304730 0.007717875    1.0000000

red - yellow  -0.12344537 0.1294790 0.908972974    1.0000000
```

```
> plot(simData$x1, predict(fit4))
```



```
> par(mfrow = c(1, 2))
> ts.plot(fit4$mcemEST, main = "MLEs estimates",
+         xlab = "EM Iteration", ylab = "MLE value")
> ts.plot(fit4$loglikeVal, main = "Likelihood value",
+         xlab = "EM Iteration", ylab = "Likelihood")
```

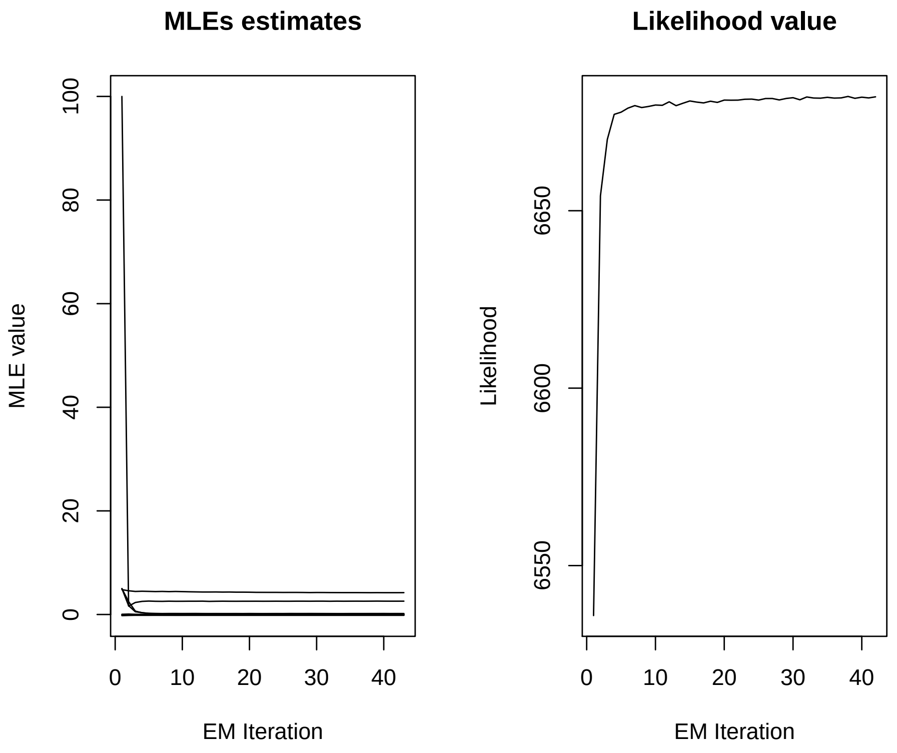


Figure 9: MLEs estimates and loglikelihood value after each EM iteration.