# Mixed Effects GLM

*Han Bossier*

*30/7/2018*

## Contents

# 1 Introduction

This report focusses on estimating the variance of the fixed effects parameters in a mixed effects general linear model. Note that the report called *mixedEffects* is focused on fMRI time series. Here we generate subjects into classes and use the mixed modeling framework to deal with this structure.

## 1.1 Source Code

Monte-Carlo simulations are run on HPC. See *mixed_effects_glm.R* and *mixed_effects_glm.sh* for original source code.

# 2 Data Generating Model

Consider the following linear model for a specific clusters $i = 1, \ldots, K$ (in our case a class which will contain subjects):

$$\mathbf{Y_i} = \mathbf{X_i}\boldsymbol{\beta} + \mathbf{Z_i}\mathbf{u_i} + \boldsymbol{\varepsilon}_i \tag{1}$$

$$\mathbf{u_i} \sim N(\mathbf{0}, \mathbf{D}) \tag{2}$$

$$\boldsymbol{\varepsilon}_i \sim N(\mathbf{0}, \sigma_w^2 I), \tag{3}$$

where:

- $\mathbf{Y_i}$ is the response vector of dimension $n_i \times 1$ where $n_i$ is the sample size in cluster/class $i$
- $\mathbf{X_i}$ is the $n_i \times p$ model matrix for the fixed effects with $p$ the number of predictors including the intercept
- $\boldsymbol{\beta}$ is the $p \times 1$ vector of fixed effects coefficients
- $\mathbf{Z_i}$ is the $n_i \times q$ model matrix for the random effects where $q$ equals the number of random effect parameters
- $\mathbf{u_i}$ is the $q \times 1$ vector of random effect coefficients
- $\boldsymbol{\varepsilon}_i$ is the $n_i \times 1$ vector of error terms

One can combine all clusters into:

$$\mathbf{Y} = \mathbf{X}\boldsymbol{\beta} + \mathbf{X}\mathbf{u} + \boldsymbol{\varepsilon} \tag{4}$$

We will generate data using one continuous predictor and two random effects (one for the intercept and one for the effect of X on Y, $q = 2$). Hence, the $q \times q$ variance-covariance matrix of the random effects ($\mathbf{D}$) equals:

$$\mathbf{D} = \begin{bmatrix} \sigma_{b0}^2 & 0 \\ 0 & \sigma_{b1}^2. \end{bmatrix} \tag{5}$$

In this report, we set $\sigma_{b0}^2 = 1$ and $\sigma_{b1}^2 = 2$. Note that we generate data with homogeneous error variances where we set $\sigma_w^2 = 1$. We will have 40 classes ($K = 40$) and 50 subjects within each class ($n_i = 50$ for all $i = 1, \ldots, K$). Furthermore, we set $\beta_0 = 5$ and $\beta 1 = 2$ in the $\boldsymbol{\beta}$ vector.

For the Monte-Carlo simulations, we will generate the entire $\mathbf{u}$ matrix beforehand. Hence in $\mathbf{R}$, we have the following true (parameter) values and $\mathbf{u}$:

```r
# Class and sample size
numclass <- 40
NinClass <- 50

# Fixed effects parameters
beta0 <- 5
beta1 <- 2

# Random effects parameters
sigma_e <- 1
sigmab0 <- 1
sigmab1 <- 2

# Generate the variance covariance matrix of the random effects
var_cov_U <- rbind(c(sigmab0**2, 0), c(0, sigmab1**2))
# Generate the values for b0 and b1
B_matrix <- MASS::mvrnorm(numclass, mu = c(0,0), Sigma = var_cov_U)
```

# 3 Variance of Fixed Effects Parameters

To estimate the variances of the fixed effect parameters ($\beta$), we first define the variance-covariance matrix of the observed responses for class $i$, $\text{Var}(\mathbf{Y}_i)$:

$$\text{Var}(\mathbf{Y}_i) = \mathbf{V_i} = \mathbf{Z_i D Z_i}' + \sigma_w^2 I \tag{6}$$

Then the variance-covariance matrix of the fixed effect parameters is given by:

$$\text{Var}(\boldsymbol{\beta}) = (\sum_{i=1}^{K} \mathbf{X}_i \mathbf{V}^{-1} \mathbf{X_i})^{-1}. \tag{7}$$

In order to calculate the true variance-covariance matrix of our fixed effect parameters, we hence need to generate all responses in each Monte-Carlo simulation run beforehand. In **R**, this is:

```r
# First we create empty vectors for X and V, as well as an empty matrix
# for the variance-covariance matrices of the fixed effects.
ComplX <- ComplV <- matrix(NA,nrow = 1, ncol = 1)
VarCovBeta_raw <- matrix(0, ncol = 2, nrow = 2)
Xlist <- Zlist <- list()

# Pre-define the true variance-covariance matrix for fixed effects parameters
for(i in 1:numclass){
  # Predictors for this class
  X <- cbind(1, round(runif(n = NinClass, min = 1, max = 20)))

  # Z-matrix for this class
  Z <- X

  # V-matrix
  V <- Z %*% var_cov_U %*% t(Z) +
```

```
    diag(sigma_e**2, NinClass)

  # Part of var-covar-beta matrix
  VarCovBeta_raw <- VarCovBeta_raw + t(X) %*% solve(V) %*% X

  # Save X and Z
  Xlist[[i]] <- X
  Zlist[[i]] <- Z
}
```

While summing in each class, we get the final true variance covariance matrix for the fixed effects parameters:

```
# Now calculate true variance-covariance matrix
VarCovBeta <- solve(VarCovBeta_raw)
VarCovBeta
```

```
              [,1]          [,2]
[1,]  0.0273495531 -0.0001763937
[2,] -0.0001763937  0.1000169024
```

```
# Standard error of beta = sqrt(var(beta))
SEBeta <- data.frame('term' = c('(Intercept)', 'X'),
                     'TrueSE' = sqrt(diag(VarCovBeta)),
                     stringsAsFactors = FALSE)
SEBeta
```

```
# A tibble: 2 x 2
  term        TrueSE
  <chr>        <dbl>
1 (Intercept)  0.165
2 X            0.316
```

# 4 Monte-Carlo Simulation

The **R** script is run on the HPC infastructure. Therefore, I show the code here for one run. We refer to the file *mixed_effects_glm.R* and *mixed_effects_glm.sh* for the original source code.

```
# Empty data frame with simulation results
FitTotDat <- data.frame() %>% as_tibble()

# For loop over ONE simulation
for(r in 1:1){
  # Set starting seed
  starting.seed <- pi*r
  set.seed(starting.seed)

  # Empty data frame
  TotDat <- data.frame()

  # Loop over the classes
  for(i in 1:numclass){
      # Loop over the subjects
      for(j in 1:NinClass){
      # Generate data using: X*beta + Z*u + e
        dat <- Xlist[[i]] %*% matrix(c(beta0, beta1), ncol = 1) +
```

```
          Zlist[[i]] %*% matrix(c(B_matrix[i,1], B_matrix[i,2]), ncol = 1) +
          rnorm(n = NinClass, mean = 0, sd = sigma_e)

      # Add to data frame
      TotDat <- data.frame(Y = dat, X = Xlist[[i]][,2], class = i) %>% as_tibble() %>%
        bind_rows(TotDat,.)
    }
  }

  # Analysis
  fit <- lmer(Y ~ 1 + X + (1 + X|class), data = TotDat, REML = TRUE)
  FitTotDat <- broom::tidy(fit) %>%
    # Add true SE
    left_join(.,SEBeta, by = 'term') %>%
    mutate(sim = r) %>%
    bind_rows(FitTotDat,.)
}
```

Note that the analysis consist of fitting a random effects intercept and slope model using *lmer* and *REML*.

For instance, we have in this Monte-Carlo simulation run:

```
head(TotDat)
```

```
# A tibble: 6 x 3
      Y     X class
* <dbl> <dbl> <int>
1  30.3 15.0      1
2  34.4 17.0      1
3  26.3 12.0      1
4  18.0  8.00     1
5  14.1  5.00     1
6  31.3 15.0      1
```

```
fit <- lmer(Y ~ 1 + X + (1 + X|class), data = TotDat, REML = TRUE)
summary(fit)
```

```
Linear mixed model fit by REML ['lmerMod']
Formula: Y ~ 1 + X + (1 + X | class)
   Data: TotDat

REML criterion at convergence: 285416.4

Scaled residuals:
    Min      1Q  Median      3Q     Max
-4.7593 -0.6731  0.0035  0.6683  4.3697

Random effects:
 Groups   Name        Variance Std.Dev. Corr
 class    (Intercept) 0.6575   0.8109
          X           3.4695   1.8627   0.20
 Residual             1.0084   1.0042
Number of obs: 100000, groups:  class, 40

Fixed effects:
            Estimate Std. Error t value
```

```
(Intercept)   5.3311     0.1284   41.52
X             1.7486     0.2945    5.94

Correlation of Fixed Effects:
  (Intr)
X 0.200
```

# 5 Analysis

We pull the data and look at the asymptotic properties after having run 1000 Monte-Carlo simulation runs (100 batches of 10 runs).

```
# Number of batches
nsim <- 100

# Empty data frame
FitTotDat <- data.frame() %>% as_tibble()

# Read in data
for(i in 1:nsim){
  FitTotDat <- readRDS(paste(RawDat, '/MixEffglm_', i, '.rda', sep = '')) %>%
    bind_rows(FitTotDat, .)
}
```

## 5.1 Fixed Effects Parameters

First we have a look at the fixed effect parameters and its estimated variances. We filter out the intercept and the predictor $X$ (i.e. the fixed effects part), then add the true parameter values for $\beta$ and summarise over Monte-Carlo simulation runs. We take the average over the estimates as well as the average estimated standard error (which is the square root of the variance of the fixed effects parameter estimates).

```
FitTotDat %>%
  filter(term %in% c('(Intercept)', 'X')) %>%
  # Add true values
  left_join(.,data.frame(term = c('(Intercept)', 'X'),
                         TrueEst = c(beta0, beta1), stringsAsFactors = FALSE),
            by = 'term') %>%
  group_by(term) %>%
  summarise(AvgEst = mean(estimate),
            TrueEst = mean(TrueEst),
            AvgSE = mean(std.error),
            TrueSE = mean(TrueSE))
```

```
# A tibble: 2 x 5
  term        AvgEst TrueEst AvgSE TrueSE
  <chr>        <dbl>   <dbl> <dbl>  <dbl>
1 (Intercept)   5.01    5.00 0.155  0.166
2 X             2.04    2.00 0.309  0.316
```

### 5.2 Random Effects Parameters

Finally, let us check the random effects parameters. We filter out the estimated standard deviations of the residuals, the random intercept and the random slope. Then we join the true values we used in the data generating model. Afterwards we average the standard deviations over all simulations.

```
FitTotDat %>%
  filter(term %in% c('sd_Observation.Residual',
                     'sd_(Intercept).class',
                     'sd_X.class')) %>%
  dplyr::select(term, estimate, sim) %>%
  left_join(.,data.frame('term' = c('sd_Observation.Residual',
                                    'sd_(Intercept).class',
                                    'sd_X.class'),
                         'TrueSD_ran' = c(sigma_e, sigmab0, sigmab1),
                         stringsAsFactors = FALSE),
            by = 'term') %>%
  group_by(term) %>%
  summarise(AvgEst = mean(estimate),
            AvgTrueSD = mean(TrueSD_ran))
```

```
# A tibble: 3 x 3
  term                    AvgEst AvgTrueSD
  <chr>                    <dbl>     <dbl>
1 sd_(Intercept).class     0.977      1.00
2 sd_Observation.Residual  1.000      1.00
3 sd_X.class               1.96       2.00
```

## 6 Conclusion

It seems pretty good (for once)! :-) Now go and generate fMRI time series using this approach!

## 7 Session

```
sessionInfo()
```

```
R version 3.4.3 (2017-11-30)
Platform: x86_64-apple-darwin15.6.0 (64-bit)
Running under: macOS High Sierra 10.13.6

Matrix products: default
BLAS: /Library/Frameworks/R.framework/Versions/3.4/Resources/lib/libRblas.0.dylib
LAPACK: /Library/Frameworks/R.framework/Versions/3.4/Resources/lib/libRlapack.dylib

locale:
[1] nl_BE.UTF-8/nl_BE.UTF-8/nl_BE.UTF-8/C/nl_BE.UTF-8/nl_BE.UTF-8

attached base packages:
[1] tcltk     stats     graphics  grDevices utils     datasets  methods
[8] base
```

```
other attached packages:
 [1] bindrcpp_0.2       fMRIGI_0.1.0       NeuRRoStat_0.1.0
 [4] neuRosim_0.2-12    devtools_1.13.4    metafor_2.0-0
 [7] mvmeta_0.4.7       RColorBrewer_1.1-2 MASS_7.3-47
[10] lme4_1.1-14        Matrix_1.2-12      reshape2_1.4.3
[13] tidyr_0.7.2        tibble_1.4.2       dplyr_0.7.4
[16] ggplot2_2.2.1.9000 oro.nifti_0.9.1    gridExtra_2.3
[19] lattice_0.20-35    AnalyzeFMRI_1.1-16 fastICA_1.2-1
[22] R.matlab_3.6.1

loaded via a namespace (and not attached):
 [1] Rcpp_0.12.15      assertthat_0.2.0  rprojroot_1.2
 [4] digest_0.6.14     psych_1.7.8       utf8_1.1.3
 [7] R6_2.2.2          plyr_1.8.4        backports_1.1.1
[10] evaluate_0.10.1   pillar_1.1.0      rlang_0.1.6.9003
[13] lazyeval_0.2.1    minqa_1.2.4       nloptr_1.0.4
[16] R.utils_2.6.0     R.oo_1.21.0       rmarkdown_1.8
[19] splines_3.4.3     foreign_0.8-69    stringr_1.2.0
[22] munsell_0.4.3     broom_0.4.3       compiler_3.4.3
[25] pkgconfig_2.0.1   mnormt_1.5-5      htmltools_0.3.6
[28] RNifti_0.7.1      crayon_1.3.4      withr_2.1.1.9000
[31] bitops_1.0-6      R.methodsS3_1.7.1 grid_3.4.3
[34] nlme_3.1-131      gtable_0.2.0      magrittr_1.5
[37] scales_0.5.0.9000 cli_1.0.0         stringi_1.1.6
[40] tools_3.4.3       glue_1.2.0        purrr_0.2.4
[43] abind_1.4-5       parallel_3.4.3    yaml_2.1.14
[46] colorspace_1.3-2  memoise_1.1.0     knitr_1.17
[49] bindr_0.1
```