Variance Estimation - Third Level FSL

Han Bossier

22/6/2018

Contents

1 Introduction

In this report, we simulate time series at subject level, combine them using a *FLAME1* (mixed model from FSL), generate several studies and finally combine the studies with both *lmer* (**R**) and *FLAME1*.

2 Simulation

In this section, we discuss the data generation and the models we use to analyze the data. This will be using a mixed model approach with *lmer* (**R**), while the second approach is using the typical two-stage approach in fMRI and the mixed model *FLAME1* from **FSL**.

2.1 Data generation

Let us first define some global variables.

```
# Location of raw data
RawDat <- '/Volumes/2_TB_WD_Elements_10B8_Han/PhD/Simulation/Results/Variance_3lvl/'
# During OHBM, I used the following path
# RawDat <- '/Users/hanbossier/Desktop/Results_VAR3LVL/'

# Number of batches simulated
IDs <- 200

# Number of studies
nstud <- 40

# Number of subjects
nsub <- 50

# Value for sigma in the model
sigma_eps <- 100

# Between study variability: intercept
sigma_eta1 <- c(0, 1, 1)

# Between study variability: slope
sigma_eta2 <- c(0, 2, 4)

# Between subject variability (variance of random slope)
sigma_b2 <- c(0, 5, 10)

# Variance of random intercept
sigma_b1 <- c(0, 1, 1)
```

We generate time series for each subject in each study for *one voxel*. We have 200 scans (T) with a TR of 2 seconds, a block design of 20 sec ON/OFF, a double-gamma HRF convolution and a true BOLD percent signal change of 3 percent. We generate T datapoints (Y) for each subject $i = 1, \ldots, N$ in study $j = 1, \ldots, K$ using the following GLM:

$$Y_{ij} = (\beta_0 + b_{0ij}) + (\beta_1 + b_{1ij})X + \varepsilon_{ij},$$

with $b_{0ij} \sim N(0, 2)$, $b_{1ij} \sim N(0, \sigma_b^2 + \sigma_s^2)$ (sum of between-subject and between-study variability) and $\varepsilon_{ij} \sim N(0, \sigma_e^2)$. We see that we use the same design matrix X for each subject and we only generate Gaussian random (white) noise. Furthermore, we have $\sigma_b^2 \in \{0, 5^2, 10^2\}$, $\sigma_s^2 \in \{0, 2^2, 4^2\}$ and $\sigma_e^2 = 10^2$. Finally, we have $\beta_1 = 3$ and $\beta_0 = 100$.

In the following, we generate an example time series:

```
# Signal characteristics
TR <- 2
nscan <- 200
total <- TR*nscan
on1 <- seq(1,total,40)
onsets <- list(on1)
duration <- list(20)

####################
#### Generate a design: GROUND TRUTH DESIGN
####################

# true %BOLD change
BOLDC <- 3

# Base/intercept of signal
intcpt <- 100

########################################
#### DESIGN AND SIGNAL TIME SERIES ####
########################################

# Generating a design matrix: convolution of block design with double-gamma HRF
X <- neuRosim::simprepTemporal(total,1,onsets = onsets,
                               effectsize = 1, durations = duration,
                               TR = TR, acc = 0.1, hrf = "double-gamma")

# X vector for one subject = predicted signal
X_s <- neuRosim::simTSfmri(design=X, base=0, SNR=1, noise="none", verbose=FALSE)

# Now the model will be: (intcpt + b1) + (BOLDC + b2) * pred + epsilon

## Design parameters
# Extend the design matrix with the intercept
xIN <- cbind(intcpt, X_s)

# Contrast: not interested in intercept
CONTRAST <- matrix(c(0,1),nrow=1)

# Calculate (X'X)^(-1) with contrast
design_factor <- CONTRAST %*% (solve(t(xIN) %*% xIN )) %*% t(CONTRAST)
```

To generate random intercept and slopes, we first generate study-specific values for $b_{0.j}$ and $b_{1.j}$ where we set $\sigma_s^2 = 2^2$ using a variance-covariance matrix:

```
# Generate var-covar matrix of studies: random intercept + slope
var_cov_Study <- rbind(c(sigma_eta1[2]**2, 0), c(0, sigma_eta2[2]**2))

# Now generate the study specific random intercept and slope values using this matrix
```

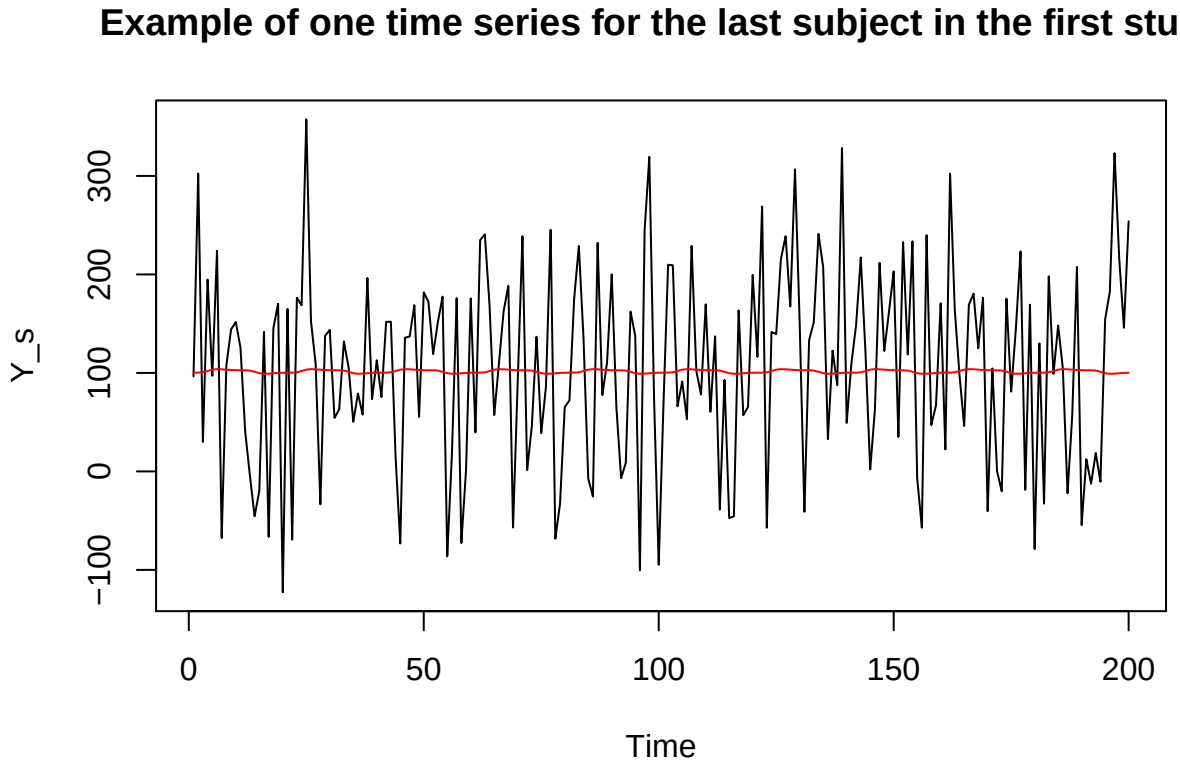
```
Stud_matrix <- MASS::mvrnorm(nstud, mu=c(0,0), Sigma = var_cov_Study)
```

Now looping over all subjects, we generate first for all subjects the variance-covariance matrix (with $\sigma_b^2 = 5^2$) and from there generate the time series for each subject.

```
# ---------------------------
# INSIDE THE STUDIES FOR LOOP
s <- 1
# ---------------------------

# Generate D matrix (subject): variance-covariance matrix of random intercept + slope
# Variance of slope = sigma_b**2
var_cov_D <- rbind(c(sigma_b1[2]**2, 0), c(0, sigma_b2[2]**2))

# Generate the subject-specific values for intercept and slope using this D-matrix
B_matrix <- MASS::mvrnorm(nsub, mu=c(0,0), Sigma = var_cov_D)

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

# For loop over all subjects
for(i in 1:nsub){
  # Generate nscan values, corresponding to time series of one subject
  # note: random intercepts and random slopes generated earlier
  Y_s <- (intcpt + Stud_matrix[s,1] + B_matrix[i,1]) +
          ((BOLDC + Stud_matrix[s,2] + B_matrix[i,2]) * X_s) +
          rnorm(n = nscan, mean = 0, sd = sigma_eps)

  # Add to data frame
  Y <- data.frame(Y = Y_s, X = X_s, sub = as.integer(i),
                  stud = as.integer(s)) %>% as_tibble() %>%
    bind_rows(Y, .)
}
```

Let us plot the time series for one subject in study (s) in one active voxel (red line is true response for this subject).

```
plot(Y_s, type = 'l', main = 'Example of one time series for the last subject in the first study',
     xlab = 'Time')
lines(x = ((intcpt + B_matrix[nsub,1] + Stud_matrix[1,1]) + ((BOLDC + B_matrix[nsub,2] + Stud_matrix[1,1
```



We combine all subjects in each study using a mixed effects model from FSL. Then we combine the maps from each study in a 4D COPE and VARCOPE *nifti* image and write this map in a temp folder. We also save all time series over all studies in the object `Y_stud`.

2.2 Models

The analysis of the time series of all N subjects using **R** is straightforward:

```
lmer(Y ~ 1 + X + (1 + X|stud/sub), data = Y_stud)
```

Using **FSL**, we split the analysis in three stages. First we fit the time series to each subject (using OLS as we generate white noise) and save the estamated parameter with its variance in *.nifti* files which we write to a temporary folder. Note, we had to create a matrix of $2 \times 2 \times 2$ voxels, all containing the same values as FSL is not able to run its functions on just one voxel (I think it has something to do with the indices). Then we run *feat* (option *flame1*) on the 4D cope and 4D varcope maps with a given mask. Note that we first generate some auxilirary files (such as the design matrix), then run *feat* (not executed here) and finally read back the results in. We repeat this procedure at the third level.

For example, the following code can be used to fit group level models.

```
# For this, we need to first analyze each subject individually, save COPE and VARCOPE
# and then proceed.
# We call this object secLevel
secLevel <- Y %>%
  group_by(sub) %>%
  do(.,
     # For each subject, fit linear model with an intercept and X as predictors
     broom::tidy(
       lm(Y ~ 1 + X, data = .))) %>%
  # Filter on predictor
  filter(term == 'X') %>%
```

```
  # Now select the estimate and standard error
  dplyr::select(sub, estimate, std.error) %>%
  # Create variance
  mutate(varCope = std.error^2)

# Create 4D images (all voxels in first 3 dimensions are the same), otherwise FSL crashes!
# Then convert the estimates and variance to nifti images
COPE4D <- nifti(img=array(rep(as.numeric(secLevel$estimate), each = 8),
                          dim=c(2,2,2,nsub)),
                dim=c(2,2,2,nsub), datatype = 16)
VARCOPE4D <- nifti(img=array(rep(as.numeric(secLevel$varCope), each = 8),
                             dim=c(2,2,2,nsub)),
                   dim=c(2,2,2,nsub), datatype = 16)

# Write them to DataWrite
writeNIfTI(COPE4D, filename = paste(DataWrite,'/COPE',sep=''), gzipped=FALSE)
writeNIfTI(VARCOPE4D, filename = paste(DataWrite,'/VARCOPE',sep=''), gzipped=FALSE)

#########################
## ETC ETC SEE var_3lvl.R
#########################

# Example of run in FSL
command <- paste(fslpath, 'flameo --cope=COPE --vc=VARCOPE --mask=mask --ld=FSL_stats --dm=design.mat
Sys.setenv(FSLOUTPUTTYPE="NIFTI")
system(command)

#########################
## ETC ETC SEE var_3lvl.R
#########################

########################
## SAME FOR THIRD LEVEL
########################
```

Note, we simulate all actual data in the var_3lvl.R script.

2.3 True Values

Before we go to the results, we first calculate the true values. We will do this for the three-stage model formulation as the variance estimation of the random effects model is complicated (it involves taking the inverse of the full observation matrix which is too large). The three-stage model for a simplified (one predictor) GLM is defined as:

$$Y_{jit} = \beta_0 + \beta_1 X + \varepsilon_{jit}, \quad i = 1, \ldots, N \quad j = 1, \ldots, S \quad \text{and} \quad t = 1, \ldots, T. \tag{1}$$

In the second stage, we get:

$$Y_G = \beta_1^* X_G + \varepsilon^*, \tag{2}$$

In the third stage, we get:

$$Y_S = \beta_1^{**} X_S + \varepsilon^{**}, \tag{3}$$

where Y_G is the vector of estimated first level parameters ($\hat{\beta}_1$) and X_G equals a column of 1's with length N. Then Y_S is the vector of estimated second level parameters ($\hat{\beta}_G$) and X_S equals a column of 1's with length S. In this case, $\varepsilon^* \sim N(0, \sigma_b^2 + \text{Var}(\widehat{\beta_1}))$. Denote σ_G^2 as $\text{Var}(\varepsilon^*)$ and note that is a mixed error component containing both variability of the estimation at the first level and a between-subject variability component σ_b^2. Then we have $\varepsilon^{**} \sim N(0, \sigma_s^2 + \text{Var}(\widehat{\beta_1^*}))$, which contains variability of the estimation at the second level and between-study variability.

Furthermore, we have:

$$\text{Var}(\widehat{\beta}_1) = \frac{\widehat{\sigma}_e^2}{\sum_{t=1}^{T}(X_t - \overline{X})^2} \tag{4}$$

Next, we have:

$$\text{Var}(\widehat{\beta}_1^*) = \frac{\sigma_b^2 + \frac{\sigma_e^2}{\sum_{t=1}^{T}(X_t - \overline{X})^2}}{N} \tag{5}$$

Finaly, we find the variance of the estimated study level parameters ($\text{Var}(\beta_1^{**})$) combining the equations from above:

$$\text{Var}(\beta_1^{**}) = \frac{\sigma_s^2 + \frac{\sigma_b^2 + \frac{\sigma_e^2}{\sum_{t=1}^{T}(X_t - \overline{X})^2}}{N}}{S} \tag{6}$$

In **R**, this is for all values of between-subject and between-study variability:

```
X_G <- matrix(1, nrow = nsub, ncol = 1)
trueVarBetaG <- c((sigma_b2^2 + (sigma_eps^2 / sum((X_s - mean(X_s))^2))) / nsub)
X_S <- matrix(1, nrow = nstud, ncol = 1)
trueVarBetaS <- (sigma_eta2 + trueVarBetaG) / nstud
trueVarBetaS
```

```
[1] 0.1271912 0.1896912 0.2771912
```

3 Results

Here, we read in the simulation results.

```
allDat <- data.frame() %>% as.tibble()
# For loop over the batches
for(i in 1:IDs){
  allDat <- bind_rows(allDat,
  readRDS(file = paste(RawDat, 'Results_bsub_',sigma_b2[1],'_bstud_',
                       sigma_eta2[1],'/VAR3LVL_',i,'.rda', sep = '')) %>%
    mutate(True_SD_bstud = sigma_eta2[1]),
```

```
  readRDS(file = paste(RawDat, 'Results_bsub_',sigma_b2[2],'_bstud_',
                       sigma_eta2[2],'/VAR3LVL_',i,'.rda', sep = '')) %>%
    mutate(True_SD_bstud = sigma_eta2[2]),
  readRDS(file = paste(RawDat, 'Results_bsub_',sigma_b2[3],'_bstud_',
                       sigma_eta2[3],'/VAR3LVL_',i,'.rda', sep = '')) %>%
    mutate(True_SD_bstud = sigma_eta2[3])
  )
}
```

We can now check:

- the average of the β_1 estimates, should be `3`
- the empirical variance of the β_1 estimates (variance over Monte-Carlo simulations)
- the average of the estimated variance of β_1^{**} (average over simulations of $Var(\beta_1^{**})$). This should approach `0.1271912, 0.1896912, 0.2771912`, depending on the true value of σ_s^2.
- the average estimated between-study variability. In FSL this is the file *mean_random_effects_var1*, with **R** this is the estimated term from *lmer*. It should approach `0, 2, 4`.
- the empirical coverage of the 95% CI around the true value of β_1.

```
allDat %>% group_by(type, True_SD_bstud) %>%
  summarise(Avg_beta = mean(estimate),
            Obs_var_beta = var(estimate),
            Avg_est_var_beta = mean(variance),
            Avg_sd_bstud = mean(SD_bsub),
            EC = mean(EC)) %>%
  mutate(TrueVar_beta = trueVarBetaS) %>%
  #Re-arrange
  ungroup() %>%
  dplyr::select(type, Avg_beta, TrueVar_beta, Obs_var_beta,
                Avg_est_var_beta, True_SD_bstud, Avg_sd_bstud, EC) %>%
  # Print to table
  kable(., caption = 'Results of Monte-Carlo simulation study', digits = 3)
```

Table 1: Results of Monte-Carlo simulation study

type	Avg_beta	TrueVar_beta	Obs_var_beta	Avg_est_var_beta	True_SD_bstud	Avg_sd_bstud	EC
FLAME	3.012	0.127	0.129	0.145	0	0.352	0.964
FLAME	3.018	0.190	0.167	0.174	2	0.878	0.961
FLAME	3.017	0.277	0.204	0.213	4	1.026	0.958
LMER	3.013	0.127	0.127	0.142	0	0.538	0.964
LMER	3.019	0.190	0.164	0.176	2	1.057	0.964
LMER	3.019	0.277	0.201	0.216	4	1.086	0.964

Not sure if the true variance between-studies is correct...!

4 Session

```
sessionInfo()
```

```
R version 3.4.3 (2017-11-30)
Platform: x86_64-apple-darwin15.6.0 (64-bit)
Running under: macOS High Sierra 10.13.5
```

```
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] knitr_1.17         tidyr_0.7.2        tibble_1.4.2
[16] dplyr_0.7.4        ggplot2_2.2.1.9000 oro.nifti_0.9.1
[19] gridExtra_2.3      lattice_0.20-35    AnalyzeFMRI_1.1-16
[22] fastICA_1.2-1      R.matlab_3.6.1

loaded via a namespace (and not attached):
 [1] purrr_0.2.4       splines_3.4.3     colorspace_1.3-2
 [4] htmltools_0.3.6   yaml_2.1.14       rlang_0.1.6.9003
 [7] R.oo_1.21.0       pillar_1.1.0      nloptr_1.0.4
[10] withr_2.1.1.9000  glue_1.2.0        R.utils_2.6.0
[13] plyr_1.8.4        bindr_0.1         stringr_1.2.0
[16] munsell_0.4.3     gtable_0.2.0      RNifti_0.7.1
[19] R.methodsS3_1.7.1 evaluate_0.10.1   memoise_1.1.0
[22] highr_0.6         Rcpp_0.12.15      scales_0.5.0.9000
[25] backports_1.1.1   abind_1.4-5       digest_0.6.14
[28] stringi_1.1.6     grid_3.4.3        rprojroot_1.2
[31] tools_3.4.3       bitops_1.0-6      magrittr_1.5
[34] lazyeval_0.2.1    pkgconfig_2.0.1   assertthat_0.2.0
[37] minqa_1.2.4       rmarkdown_1.8     R6_2.2.2
[40] nlme_3.1-131      compiler_3.4.3
```