# Printed Script: Qian et al. (2020)

**Script: Generative Model**

```
# 2018.10.22 Tianchen Qian

# generative model:
# Y_t+1 = alpha_0 + alpha_1 X_t + b_0i + b_1i X_t + A_t (beta_0 + beta_1 X_t + b_2i + b_3i X_t) +
# epsilon_it

dgm_with_treatment <- function(sample_size, total_T, dgm_type) {

    # dgm_type is in c(1,2,3,4)
    stopifnot(dgm_type %in% c(1,2,3,4))

    # dgm_type = 1 or 3
    alpha_0 <- - 1
    alpha_1 <- - 0.3
    beta_0 <- 0.5
    beta_1 <- 0.1
    sigma_b0 <- 2
    sigma_b1 <- 0
    sigma_b2 <- 1
    sigma_b3 <- 0
    sigma_eps <- 1

    if (dgm_type == 2) {
        sigma_b1 <- sigma_b3 <- 0.5
    }
    if (dgm_type == 4) {
        sigma_b2 <- 0
    }


    prob_a <- 0.5

    df_names <- c("userid", "day", "X", "prob_A", "A", "Y", "b0", "b1", "b2", "b3", "eps", "delta")

    dta <- data.frame(matrix(NA, nrow = sample_size * total_T, ncol = length(df_names)))
    names(dta) <- df_names

    dta$userid <- rep(1:sample_size, each = total_T)
    dta$day <- rep(1:total_T, times = sample_size)

    # uncorrelated random effects
    b_0i <- rnorm(sample_size, mean = 0, sd = sigma_b0)
    b_1i <- rnorm(sample_size, mean = 0, sd = sigma_b1)
    b_2i <- rnorm(sample_size, mean = 0, sd = sigma_b2)
    b_3i <- rnorm(sample_size, mean = 0, sd = sigma_b3)

    # b_1i[b_1i > 2] <- 2
    # b_1i[b_1i < -2] <- -2
    #
    # b_3i[b_3i > 2] <- 2
    # b_3i[b_3i < -2] <- -2

    for (t in 1:total_T) {
```

```
        # row index for the rows corresponding to day t for every subject
        row_index <- seq(from = t, by = total_T, length = sample_size)

        if (dgm_type == 1) {
            if (t == 1) {
                dta$X[row_index] <- rnorm(sample_size)
            } else {
                dta$X[row_index] <- dta$Y[row_index_lag1] + rnorm(sample_size)
            }
            dta$prob_A[row_index] <- rep(prob_a, sample_size)
        } else if (dgm_type == 2) {
            if (t == 1) {
                dta$X[row_index] <- rnorm(sample_size)
            } else {
                dta$X[row_index] <- dta$Y[row_index_lag1] + rnorm(sample_size)
            }
            dta$prob_A[row_index] <- ifelse(dta$X[row_index] > - 1.27, 0.7, 0.3)
        } else if (dgm_type %in% c(3,4)) {
            if (t == 1) {
                dta$X[row_index] <- rnorm(sample_size, mean = b_0i) # X involves b_i!!
            } else {
                dta$X[row_index] <- dta$Y[row_index_lag1] + rnorm(sample_size, mean = b_0i)
                # X involves b_i!!
            }
            dta$prob_A[row_index] <- rep(prob_a, sample_size)
        }

        dta$A[row_index] <- rbinom(sample_size, 1, dta$prob_A[row_index])

        dta$eps[row_index] <- rnorm(sample_size, mean = 0, sd = sigma_eps)

        dta$delta[row_index] <- beta_0 + beta_1 * dta$X[row_index] + b_2i + b_3i * dta$X[row_index]

        dta$Y[row_index] <- alpha_0 + alpha_1 * dta$X[row_index] + b_0i + b_1i * dta$X[row_index] +
            dta$A[row_index] * dta$delta[row_index] + dta$eps[row_index]

        dta$b0[row_index] <- b_0i
        dta$b1[row_index] <- b_1i
        dta$b2[row_index] <- b_2i
        dta$b3[row_index] <- b_3i

        row_index_lag1 <- row_index
    }

    return(dta)
}


##### example: use of lmer() #####
if( 0 ){
    sample_size <- 1000
    total_T <- 20

    dta <- dgm_with_treatment(sample_size, total_T, dgm_type = 1)
    summary(dta)
    # dta$A <- dta$A - dta$prob_A # action centering doesn't matter when prob_A is constant
```

```
    fit <- lmer(Y ~ X * A + (1 + A | userid), data = dta)

    fit <- lmer(Y ~ X * A + (X * A | userid), data = dta)
    fit

    summary(fit)$coefficients

    attr(summary(fit)$varcor$userid, "stddev") # estimated standard deviation of random effect
}
```

**Script: Simulation**

```
# 2018.10.22 Tianchen Qian

# simulation study for the random effects model paper (Statistical Science)

# generative model:
# Y_t+1 = alpha_0 + alpha_1 X_t + b_0i + b_1i X_t + A_t (beta_0 + beta_1 X_t + b_2i + b_3i X_t) +
# epsilon_it


rm(list = ls())
set.seed(123)

library(rootSolve)
library(geepack)
library(lme4)
library(mvtnorm)
library(foreach)
library(doMC)
library(doRNG)


source("generative_model.R")




# simulation: using lmer() packag ----------------------------------------


# if(0) {

max_cores <- 16
registerDoMC(min(detectCores() - 1, max_cores))

set.seed(120)


nsim <- 1000

# full simulation in Appendix B
design <- expand.grid(sample_size = c(30, 50, 100, 200), total_T = c(10, 20, 30), dgm_type = 1:3)

# simulation in Section 4
design <- expand.grid(sample_size = c(30, 100, 200), total_T = c(10, 30), dgm_type = 1:3)

design <- design[order(design$dgm_type), ]

for (idesign in 1:nrow(design)) {
    sample_size <- design$sample_size[idesign]
    total_T <- design$total_T[idesign]
    dgm_type <- design$dgm_type[idesign]

    # create template output structure for simulated trials with estimation error
```

```
    dta <- dgm_with_treatment(sample_size, total_T, dgm_type = dgm_type)
    if (dgm_type %in% c(1,3,4)) {
        fit <- lmer(Y ~ X * A + (1 + A| userid), data = dta)
    } else if (dgm_type == 2) {
        fit <- lmer(Y ~ X * A + (X * A| userid), data = dta)
    }
    coef <- summary(fit)$coefficients
    varcor <- summary(fit)$varcor
    for (irow in 1:nrow(coef)) {
        for (icol in 1:ncol(coef)) {
            coef[irow, icol] <- NA
        }
    }
    for (irow in 1:nrow(varcor$userid)) {
        for (icol in 1:ncol(varcor$userid)) {
            varcor$userid[irow, icol] <- NA
        }
    }
    attr(varcor, "sc") <- NA
    coef_NA_fill <- coef
    varcor_NA_fill <- varcor

    # start parallel jobs
    writeLines(c(""), "log.txt")
    sink("log.txt", append=FALSE)
    set.seed(123)
    result <- foreach(isim = 1:nsim, .combine = "c") %dorng% {
        if (isim %% 10 == 0) {
            cat(paste("Starting iteration",isim,"\n"))
        }
        dta <- dgm_with_treatment(sample_size, total_T, dgm_type = dgm_type)

        solution <- tryCatch(
            {
                if (dgm_type == 1 | dgm_type == 3) {
                    fit <- lmer(Y ~ X * A + (1 + A| userid), data = dta)
                } else if (dgm_type == 2) {
                    fit <- lmer(Y ~ X * A + (X * A| userid), data = dta)
                }
                list(coef = summary(fit)$coefficients, varcor = summary(fit)$varcor)
            },
            error = function(cond) {
                message("\nCatched error in lmer():")
                message(cond)
                return(list(coef = coef_NA_fill, varcor = varcor_NA_fill))
            })

        output <- list(solution)
    }
    sink()

    dir.create("simulation_result", showWarnings = FALSE)
    saveRDS(result, file = paste0("simulation_result/", idesign, ".RDS"))
}
```

```r
# collect results ------------------------------------------------------------



varcomp_result <- design

varcomp_result$sigma_b0_bias <- varcomp_result$sigma_b0_sd <-
    varcomp_result$sigma_b1_bias <- varcomp_result$sigma_b1_sd <-
    varcomp_result$sigma_b2_bias <- varcomp_result$sigma_b2_sd <-
    varcomp_result$sigma_b3_bias <- varcomp_result$sigma_b3_sd <-
    varcomp_result$sigma_eps_bias <- varcomp_result$sigma_eps_sd <- NA


design$alpha_0_bias <- design$alpha_0_sd <- design$alpha_0_cp <-
    design$alpha_1_bias <- design$alpha_1_sd <- design$alpha_1_cp <-
    design$beta_0_bias <- design$beta_0_sd <- design$beta_0_cp <-
    design$beta_1_bias <- design$beta_1_sd <- design$beta_1_cp <- NA

for (idesign in 1:nrow(design)) {
    result <- readRDS(paste0("20181025_simulation for paper_result/", idesign, ".RDS"))

    dgm_type <- design$dgm_type[idesign]
    # dgm_type = 1 or 3
    alpha_0_true <- - 1
    alpha_1_true <- - 0.3
    beta_0_true <- 0.5
    beta_1_true <- 0.1
    sigma_b0_true <- 2
    sigma_b1_true <- 0
    sigma_b2_true <- 1
    sigma_b3_true <- 0
    sigma_eps_true <- 1

    if (dgm_type == 2) {
        sigma_b1_true <- sigma_b3_true <- 0.5
    }
    if (dgm_type == 4) {
        sigma_b2_true <- 0
    }

    alpha_0 <- sapply(result, function(l) l$coef["(Intercept)", "Estimate"])
    alpha_0_sd <- sapply(result, function(l) l$coef["(Intercept)", "Std. Error"])
    alpha_0_df <- sapply(result, function(l) l$coef["(Intercept)", "df"])
    alpha_1 <- sapply(result, function(l) l$coef["X", "Estimate"])
    alpha_1_sd <- sapply(result, function(l) l$coef["X", "Std. Error"])
    alpha_1_df <- sapply(result, function(l) l$coef["X", "df"])
    beta_0 <- sapply(result, function(l) l$coef["A", "Estimate"])
    beta_0_sd <- sapply(result, function(l) l$coef["A", "Std. Error"])
    beta_0_df <- sapply(result, function(l) l$coef["A", "df"])
    beta_1 <- sapply(result, function(l) l$coef["X:A", "Estimate"])
    beta_1_sd <- sapply(result, function(l) l$coef["X:A", "Std. Error"])
    beta_1_df <- sapply(result, function(l) l$coef["X:A", "df"])
```

```
varcor <- lapply(result, function(l) l$varcor$userid) # variance matrix for random effects;
# this is "G"

sigma_eps <- sapply(result, function(l) attr(l$varcor, "sc"))

quantiles <- qt(0.975, alpha_0_df)
design$alpha_0_bias[idesign] <- mean(alpha_0) - alpha_0_true
design$alpha_0_sd[idesign] <- sd(alpha_0)
design$alpha_0_cp[idesign] <- mean((alpha_0_true < alpha_0 + quantiles * alpha_0_sd) &
                                   (alpha_0_true > alpha_0 - quantiles * alpha_0_sd))
# design$alpha_0_cp[idesign] <- mean((alpha_0_true < alpha_0 + 1.96 *
# design$alpha_0_sd[idesign]) & (alpha_0_true > alpha_0 - 1.96 * design$alpha_0_sd[idesign]))

quantiles <- qt(0.975, alpha_1_df)
design$alpha_1_bias[idesign] <- mean(alpha_1) - alpha_1_true
design$alpha_1_sd[idesign] <- sd(alpha_1)
design$alpha_1_cp[idesign] <- mean((alpha_1_true < alpha_1 + quantiles * alpha_1_sd) &
                                   (alpha_1_true > alpha_1 - quantiles * alpha_1_sd))
# design$alpha_1_cp[idesign] <- mean((alpha_1_true < alpha_1 + 1.96 *
# design$alpha_1_sd[idesign]) & (alpha_1_true > alpha_1 - 1.96 * design$alpha_1_sd[idesign]))

quantiles <- qt(0.975, beta_0_df)
design$beta_0_bias[idesign] <- mean(beta_0) - beta_0_true
design$beta_0_sd[idesign] <- sd(beta_0)
design$beta_0_cp[idesign] <- mean((beta_0_true < beta_0 + quantiles * beta_0_sd) &
                                  (beta_0_true > beta_0 - quantiles * beta_0_sd))
# design$beta_0_cp[idesign] <- mean((beta_0_true < beta_0 + 1.96 *
# design$beta_0_sd[idesign]) & (beta_0_true > beta_0 - 1.96 * design$beta_0_sd[idesign]))

quantiles <- qt(0.975, beta_1_df)
design$beta_1_bias[idesign] <- mean(beta_1) - beta_1_true
design$beta_1_sd[idesign] <- sd(beta_1)
design$beta_1_cp[idesign] <- mean((beta_1_true < beta_1 + quantiles * beta_1_sd) &
                                  (beta_1_true > beta_1 - quantiles * beta_1_sd))
# design$beta_1_cp[idesign] <- mean((beta_1_true < beta_1 + 1.96 *
# design$beta_1_sd[idesign]) & (beta_1_true > beta_1 - 1.96 * design$beta_1_sd[idesign]))

varcor_mean <- apply(simplify2array(varcor), 1:2, mean)
varcor_sd <- apply(simplify2array(varcor), 1:2, sd)
if (dgm_type == 1 | dgm_type == 3) {
    varcomp_result$sigma_b0_bias[idesign] <- varcor_mean[1,1] - sigma_b0_true^2
    varcomp_result$sigma_b0_sd[idesign] <- varcor_sd[1,1]
    varcomp_result$sigma_b2_bias[idesign] <- varcor_mean[2,2] - sigma_b2_true^2
    varcomp_result$sigma_b2_sd[idesign] <- varcor_sd[2,2]
} else if (dgm_type == 2) {
    varcomp_result$sigma_b0_bias[idesign] <- varcor_mean[1,1] - sigma_b0_true^2
    varcomp_result$sigma_b0_sd[idesign] <- varcor_sd[1,1]
    varcomp_result$sigma_b1_bias[idesign] <- varcor_mean[2,2] - sigma_b1_true^2
    varcomp_result$sigma_b1_sd[idesign] <- varcor_sd[2,2]
    varcomp_result$sigma_b2_bias[idesign] <- varcor_mean[3,3] - sigma_b2_true^2
    varcomp_result$sigma_b2_sd[idesign] <- varcor_sd[3,3]
    varcomp_result$sigma_b3_bias[idesign] <- varcor_mean[4,4] - sigma_b3_true^2
    varcomp_result$sigma_b3_sd[idesign] <- varcor_sd[4,4]
}
varcomp_result$sigma_eps_bias[idesign] <- mean(sigma_eps) - sigma_eps_true^2
```

```
    varcomp_result$sigma_eps_sd[idesign] <- sd(sigma_eps)
}



design <- design[order(design$dgm_type, design$total_T, design$sample_size), ]

varcomp_result <- varcomp_result[order(varcomp_result$dgm_type, varcomp_result$total_T,
                                       varcomp_result$sample_size), ]



# make LaTeX tables ----------------------------------------------------

library(xtable)

reorder_col <- c(1:3, 9:7, 6:4, 15:13, 12:10)
design_reorder <- design[, reorder_col]
print(xtable(design_reorder, digits = c(0, 0, 0, 0, rep(3, 12))),
      include.rownames = FALSE, hline.after = c(-1, 0, seq(from = 4,
                                                           to = nrow(design_reorder), by = 4)))


reorder_col <- c(1:3, 9:8, 7:6, 13:12, 11:10, 5:4)
varcomp_result_reorder <- varcomp_result[, reorder_col]
print(xtable(varcomp_result_reorder, digits = c(0, 0, 0, 0, rep(3, 10))),
      include.rownames = FALSE, hline.after = c(-1, 0, seq(from = 4,
                                                           to = nrow(varcomp_result_reorder), by = 4)))
```