Personalize treatment for longitudinal data using unspecified random-effects model

Hyunkeun Cho, Peng Wang, and Annie Qu

Western Michigan University, University of Cincinnati and University of Illinois at Urbana-Champaign

Supplementary Material

```
library(mvtnorm)
library(rpart)
library(randomForest)
library(e1071)

n_sim=2;
n_sub = 150; #number of subjects
n_obs = c(rep(10,n_sub*7/10),rep(8,n_sub*3/10))
id = rep((1:n_sub),n_obs)
total = length(id)
a = 0.8;#correlation coefficient
corr_true='ar1'
sigma=correlation(corr_true,a,n_obs)#correlation matrix

b_pqif1=b_pqif0=b_total1=b_total0=array(0, dim=c(n_sub,n_sim))
be_p0=be_p1=array(0,dim=c(2,n_sim))
COV1 = COV0 = array(0, dim=c(max(n_obs),max(n_obs),n_sim))
x11=x21=x31=x41=x51=x61=array(0,dim=c(n_sub,n_sim))
x10=x20=x30=x40=x50=x60=array(0,dim=c(n_sub,n_sim))
x=z=rep(1,n_sub*max(n_obs))
Y_total1 = Y_total0 = array(0,dim=c(total,n_sim))
x1=x2=x3=x4=x5=x6=array(0,dim=c(n_sub,n_sim))
b_1=b_0=b_1_0=array(0,dim=c(n_sub,n_sim))

############################
#Generate the simulated data#
############################
for(j in 1:n_sim){

x1[,j]=matrix(runif(n_sub,-1,1),ncol=1)
x2[,j]=matrix(runif(n_sub,-1,1),ncol=1)
```

```
x3[,j]=matrix(runif(n_sub,-1,1),ncol=1)
x4[,j]=matrix(runif(n_sub,-1,1),ncol=1)
x5[,j]=matrix(runif(n_sub,-1,1),ncol=1)
x6[,j]=matrix(runif(n_sub,-1,1),ncol=1)

b_0[,j]=x1[,j]
b_1[,j]=x2[,j]
b_1_0[,j]=b_1[,j]-b_0[,j]

x11[,j]=matrix(runif(n_sub,-1,1),ncol=1)
x21[,j]=matrix(runif(n_sub,-1,1),ncol=1)
x31[,j]=matrix(runif(n_sub,-1,1),ncol=1)
x41[,j]=matrix(runif(n_sub,-1,1),ncol=1)
x51[,j]=matrix(runif(n_sub,-1,1),ncol=1)
x61[,j]=matrix(runif(n_sub,-1,1),ncol=1)

x10[,j]=matrix(runif(n_sub,-1,1),ncol=1)
x20[,j]=matrix(runif(n_sub,-1,1),ncol=1)
x30[,j]=matrix(runif(n_sub,-1,1),ncol=1)
x40[,j]=matrix(runif(n_sub,-1,1),ncol=1)
x50[,j]=matrix(runif(n_sub,-1,1),ncol=1)
x60[,j]=matrix(runif(n_sub,-1,1),ncol=1)

b_total0[,j]=x10[,j]
b_total1[,j]=x21[,j]

     error1 = array(0, dim=c(total,1))
     error0 = array(0, dim=c(total,1))
for (i in 1:n_sub)
{
index.i=(sum(n_obs[0:(i-1)])+1):sum(n_obs[0:i])
error1[index.i,1]=mvrnorm(1,rep(0,n_obs[i]), sigma[1:n_obs[i],1:n_obs[i]])
error0[index.i,1]=mvrnorm(1,rep(0,n_obs[i]), sigma[1:n_obs[i],1:n_obs[i]])
}
      Y_total1[,j]<-rep(b_total1[,j],each=max(n_obs))+error1
      Y_total0[,j]<-rep(b_total0[,j],each=max(n_obs))+error0
Y1 = Y_total1[,j]
Y0 = Y_total0[,j]
x = as.matrix(x)

COV1[,,j]=round(cor(matrix(Y_total1[,j],n_sub,max(n_obs),byrow=TRUE)),2)
COV0[,,j]=round(cor(matrix(Y_total0[,j],n_sub,max(n_obs),byrow=TRUE)),2)
}

##########################################
#Estimate random effects and fixed effects#
```

```
##########################################
obs_ind=array(1, dim=c(n_sub*max(n_obs),1))

for(i in 1:n_sim){
Y_full1 <- array(0., dim=c(length(obs_ind)))
Y_full0 <- array(0., dim=c(length(obs_ind)))
X <- array(1, dim=c(length(obs_ind),1))
Z <- array(1, dim=c(length(obs_ind),1))
obs_linear_ind=(1:length(obs_ind))[obs_ind==1]
unobs_linear_ind=(1:length(obs_ind))[obs_ind==0]

Y_full1[obs_linear_ind]=Y_total1[,i]
Y_full0[obs_linear_ind]=Y_total0[,i]

Y1=matrix(Y_full1, byrow=TRUE, ncol=max(n_obs))
Y0=matrix(Y_full0, byrow=TRUE, ncol=max(n_obs))

n_rnd=dim(Z)[[2]]
max_n_obs=dim(Y1)[[2]]

#Get the design matrix for the random effects
proj= projection(n_sub,max_n_obs,n_rnd,Z,X)

iter_max=100
beta_new = c(0,0)
b_new1 = b_total1[,i]
b_new0 = b_total0[,i]
beta_old = beta_new;
b_old1 = b_new1;
b_old0 = b_new0;
diff = 1
iteration = 0

while (diff > 0.0000001 && iteration < iter_max) {

random <- solve_random(Y_full1,X,proj$z_design,beta_new,b_new1,
family='gaussian',log(n_sub),iter_max, unobs_linear_ind, proj$Pa,
n_sub, n_rnd, max_n_obs, type=COV1[,,i])
fixed <- solve_fix(Y_full1,X,Z,beta_new,random$b_new,family='gaussian',
"ar1",iter_max,obs_linear_ind, n_sub, max_n_obs)

diff=mean((fixed$beta-beta_old)^2)+mean((random$b_new-b_old1)^2);
beta_old=fixed$beta;
b_old1=random$b_new;
beta_new=fixed$beta;
b_new1=random$b_new;
```

```
iteration <- iteration + 1
}
be_p1[,i]=beta_new
b_pqif1[,i] = random$b_new

beta_new = c(0,0)
b_new1 = b_total1[,i]
b_new0 = b_total0[,i]
beta_old = beta_new;
b_old1 = b_new1;
b_old0 = b_new0;
diff = 1
iteration = 0

while (diff > 0.0000001 && iteration < iter_max) {

random <- solve_random(Y_full0,X,proj$z_design,beta_new,b_new0,
family='gaussian',log(n_sub),iter_max, unobs_linear_ind, proj$Pa,
n_sub, n_rnd, max_n_obs, type=COV0[,,i])
fixed <- solve_fix(Y_full0,X,Z,beta_new,random$b_new,family='gaussian',
"ar1",iter_max,obs_linear_ind, n_sub, max_n_obs)

diff=mean((fixed$beta-beta_old)^2)+mean((random$b_new-b_old1)^2);
beta_old=fixed$beta;
b_old1=random$b_new;
beta_new=fixed$beta;
b_new1=random$b_new;
iteration <- iteration + 1
}
be_p0[,i]=beta_new
b_pqif0[,i] = random$b_new
}

############################
#Build the prediction models#
############################
be.1=be_p1;be.0=be_p0;b.data1 = b_pqif1;b.data0 = b_pqif0
for(k in 1:n_sim){
b.true1=b_total1[,k]
b.true0=b_total0[,k]
treedata1=data.frame(cbind(b.data1[,k,j],x11[,k],x21[,k],x31[,k],x41[,k],
x51[,k],x61[,k] ))
treedata0=data.frame(cbind(b.data0[,k,j],x10[,k],x20[,k],x30[,k],x40[,k],
x50[,k],x60[,k] ))
```

```
test1=data.frame(cbind(b_1[,k],x1[,k],x2[,k],x3[,k],x4[,k],x5[,k],x6[,k] ))
test0=data.frame(cbind(b_0[,k],x1[,k],x2[,k],x3[,k],x4[,k],x5[,k],x6[,k] ))

#CART
fit.tree1 = rpart(X1 ~ ., data=treedata1)
fit.tree0 = rpart(X1 ~ ., data=treedata0)
tree.pred1 = predict(fit.tree1, test1)
tree.pred0 = predict(fit.tree0, test0)

#randomForest
fit.rf1=randomForest(X1 ~ ., data=treedata1)
fit.rf0=randomForest(X1 ~ ., data=treedata0)
rf.pred1=predict(fit.rf1, test1)
rf.pred0=predict(fit.rf0, test0)

#SVM
fit.svm1=svm(X1~., data=treedata1)
fit.svm0=svm(X1~., data=treedata0)
svm.pred1=predict(fit.svm1, test1)
svm.pred0=predict(fit.svm0, test0)
}

##########################################
#Function for the true workding correlation#
##########################################
correlation = function(corr_true,a,n_obs){
#correlation structure
if(corr_true=='ind'){
sigma = diag(max(n_obs))}

if(corr_true=='exch'){
sigma <- diag(max(n_obs))
m1 <- matrix(rep(1, max(n_obs) * max(n_obs)), max(n_obs)) - sigma
sigma <- sigma + a*m1}

if(corr_true=='ar1'){
sigma <- diag(max(n_obs))
for(i in 1:(max(n_obs)-1))
{
m1 <- matrix(rep(0, max(n_obs) * max(n_obs)), max(n_obs))
for (k in 1:max(n_obs)) {
     for (l in 1:max(n_obs)) {
             if (abs(k - l) == i)
                  m1[k, l] <- a^i
}
}
```

```
sigma<-sigma+m1
}
sigma <-sigma
}
return(sigma)
}

#########################################################
#Function of the design matrix for the random effects#
#########################################################
projection = function(n_sub,n_obs,n_rnd,Z,X){
z_design=array(0., dim=c(n_sub*n_obs, n_sub*n_rnd))
loc1 <- 0
loc2 <- 0
for(i in 1:n_sub){
loc1 <- loc2 + 1
loc2 <- loc1 + n_obs - 1
z_design[loc1:loc2, (0:(n_rnd-1))*n_sub+i ]=Z[loc1:loc2,]
}

#Get the projection matrix P_a
proj=X%*%solve(t(X)%*%X)%*%t(X)
null=Null(t((diag(dim(X)[[1]])-proj)%*%z_design))
Pa=null%*%solve(t(null)%*%null)%*%t(null)
Pa[abs(Pa)<0.1^9]=0
if(sum(Pa)==0){Pa=array(0., dim=c(n_sub*n_rnd, n_sub*n_rnd))}
return(list(Pa=Pa,z_design=z_design))
}

##########################################
#Function for estimation of random effects#
##########################################
solve_random <- function(Y_full,X,z_design,beta_new,b_new,family,lambda,
iter_max, unobs_linear_ind, Pa, n_sub, n_rnd, n_obs, type){

Y_full=array(Y_full, dim=c(n_sub*n_obs,1))
b_initial=array(b_new, dim=c(n_rnd*n_sub,1));
cor_mat = kronecker(diag(n_sub),solve(type))
diff <- 1
iteration <- 0
while (diff > 1e-2 && iteration < iter_max) {

    b_old=b_initial;
    u_vector=X%*%beta_new+z_design%*%b_initial;
    u_vector[unobs_linear_ind]=0;
    residual=Y_full-u_vector;
```

```
    gr  = t(z_design)%*%cor_mat%*%residual;
    gr1 = -t(z_design)%*%cor_mat%*%z_design;
    extend_score=rbind(gr, lambda*b_initial,  lambda*Pa%*%b_initial)
    extend_first_deriv=rbind(gr1, lambda*diag(n_sub*n_rnd), lambda*Pa);
    dh=t(extend_first_deriv)%*%extend_score;
    ddh=t(extend_first_deriv)%*%extend_first_deriv;
    b_initial=b_initial-solve(ddh)%*%dh;
    diff = sum(abs(b_old-b_initial));
    iteration = iteration + 1
}
if (iteration==iter_max){print('Random effects not converging!')}
return(list(b_new=array(b_initial,dim=c(n_sub,n_rnd))))
}

##########################################
#Function for estimation of fixed effects#
##########################################
solve_fix <- function(Y_full,X,Z,beta_initial,b_initial, family,
corr_structure,iter_max, obs_linear_ind, n_sub, n_obs)
{
p = dim(X)[[2]]
no_basis=1;
if(corr_structure=="exch"){
M = matrix(rep(1, n_obs * n_obs), n_obs) - diag(n_obs)
no_basis=2}
if (corr_structure=="ar1"){
M = matrix(rep(0, n_obs * n_obs), n_obs)
for (k in 1:n_obs) {
for (l in 1:n_obs) {
if (abs(k - l) == 1)
M[k, l] = 1 } }
no_basis=2
}
Y_full = array(Y_full,dim=c(n_sub*n_obs,1))
b = array(rep(b_initial,each=n_obs), dim=c(n_sub*n_obs,dim(Z)[[2]]))

beta_new=beta_initial;
diff <- 1
iteration <- 0
while (diff > 1e-5 && iteration < iter_max) {

  sumg <- matrix(rep(0, no_basis * p), nrow = no_basis * p)
  sumc <- matrix(rep(0, no_basis * p * no_basis * p), nrow = no_basis * p)
  arsumg <- matrix(rep(0, no_basis * p), nrow = no_basis * p)
  arsumc <- matrix(rep(0, no_basis * p * no_basis* p), nrow = no_basis* p)
  gi <- matrix(rep(0, no_basis * p), nrow = no_basis * p)
```

```
  arsumgfirstdev <- matrix(rep(0, no_basis * p * p), nrow = no_basis * p)
  firstdev <- matrix(rep(0, no_basis * p * p), nrow = no_basis * p)

  beta_old=beta_new;
        loc1 <- 0
        loc2 <- 0
        for (i in 1:n_sub) {
            loc1 <- loc2 + 1
            loc2 <- loc1 + n_obs - 1
            yi <- as.matrix(Y_full[loc1:loc2, ])
            xi <- X[loc1:loc2, ]
ri <- Z[loc1:loc2, ]
            bi <- as.matrix(b[loc1:loc2, ])
                ui <- xi %*% beta_new + rowSums(ri*bi)
                fui <- ui
                fui_dev <- diag(n_obs)
                vui <- diag(n_obs)

            if (corr_structure == "indep") {
                wi <- t(xi) %*% fui_dev %*% vui %*% vui
                gi0 <- wi %*% (yi - ui)
                gi[1:p, ] <- gi0
                arsumc <- arsumc + gi %*% t(gi)
                arsumg <- arsumg + gi
                di0 <- -1 * wi %*% fui_dev %*% xi
                firstdev[1:p, ] <- di0
                arsumgfirstdev <- arsumgfirstdev + firstdev
            }
            if (corr_structure != "indep"){
                wi <- t(xi) %*% fui_dev %*% vui %*% vui
                zi <- t(xi) %*% fui_dev %*% vui %*% M %*% vui
                gi0 <- wi %*% (yi - ui)
                gi1 <- zi %*% (yi - ui)
                gi[1:p, ] <- gi0
                gi[(p + 1):(2 * p), ] <- gi1
                arsumc <- arsumc + gi %*% t(gi)
                arsumg <- arsumg + gi
                di0 <- -1 * wi %*% fui_dev %*% xi
                di1 <- -1 * zi %*% fui_dev %*% xi
                firstdev[1:p, ] <- di0
                firstdev[(p + 1):(2 * p), ] <- di1
                arsumgfirstdev <- arsumgfirstdev + firstdev
            }
        }
        arcinv = solve(arsumc)
        QIF <- t(arsumg) %*% arcinv %*% arsumg
```

```
        QIF_first_dev <- t(arsumgfirstdev) %*% arcinv %*% arsumg
        QIF_second_dev <- t(arsumgfirstdev) %*% arcinv %*% arsumgfirstdev
        invarqif2dev <- solve(QIF_second_dev)
        beta_new <- beta_new - invarqif2dev %*% QIF_first_dev
        diff <- sum(abs(beta_new - beta_old))
        iteration <- iteration + 1
    }
if(iteration==iter_max){ print('Fixed effects are not converging!!')}
return(list(beta=beta_new,cov=solve(QIF_second_dev),QIF=QIF))
}
```