QLS

Jinyu Luo

2024-04-29

```
matched_samples <- readRDS(paste(path, "matched_samples.rds", sep="/"))
```

```
temp <- geeglm(y ~ bav*visit, id=pid,
                   data = matched_samples,  family = binomial('logit'),
                   corstr = "exchangeable", scale.fix = TRUE)
summary(temp)$corr
```

```
##        Estimate   Std.err
## alpha 0.1220481 0.1105247
```

GEE

```
# 1. Exchangeable
gee_ex <- geeglm(y ~ bav*visit, id=pid,
                   data = matched_samples,  family = binomial('logit'),
                   corstr = "exchangeable", scale.fix = TRUE)
# 2. Independence
gee_ind <- geeglm(y ~ bav*visit, id=pid,
                  data = matched_samples, family = binomial('logit'),
                  corstr = "independence", scale.fix = TRUE)

# 3. AR1
# M-dependence requires at least two measurements
gee_ar1 <- geeglm(y ~ bav*visit, id=pid,
                  data = matched_samples, family = binomial('logit'),
                  corstr = "ar1", scale.fix = TRUE)
```

```
N <- length(unique(matched_samples$pid))
p <- 4
DFC <- N/(N-p) # DFC stands for degrees of freddom correction
gee_results = tibble(params = c("$\\beta_0$ (Intercept)", "$\\beta_1$ (BAV)",
                                "$\\beta_2$ (Visit)",
                                "$\\beta_3$ (Visit$\\times$BAV)",
                                "$\\alpha$"),
                     ind_est = c(gee_ind$coefficients, NA),
                     ind_se = c(sqrt(diag(vcov(gee_ind))), NA), # Robust S.E.
                     exch_est = c(gee_ex$coefficients, summary(gee_ex)$corr[[1]]),
                     exch_se = c(sqrt(diag(vcov(gee_ex))),
                                 summary(gee_ex)$corr[[2]]), # Robust S.E.
```

```
                    ar1_est = c(gee_ar1$coefficients, summary(gee_ar1)$corr[[1]]),
                    ar1_se =c(sqrt(diag(vcov(gee_ar1))),
                              summary(gee_ar1)$corr[[1]])) %>%  # Robust S.E.
  mutate(ind_DFse = DFC*ind_se, exch_DFse = DFC*exch_se, ar1_DFse = DFC*ar1_se) %>%
  relocate(ind_DFse, .after = ind_se) %>%
  relocate(exch_DFse, .after = exch_se) %>%
  relocate(ar1_DFse, .after = ar1_se)
gee_results$exch_DFse[5] <- NA
gee_results$ar1_DFse[5] <- NA
```

QLS

Let $F_i(\Gamma)$ be the working correlation matrix of Y_i with the working correlation parameter $\Gamma' = (\tau', \alpha')$ where τ' represents the correlation **within matched pairs** and α' denotes the correlation between longitudinal outcomes.

$$F_i(\Gamma) = C_i(\tau) \otimes R_i(\alpha)$$

where $o\times$ denotes the Kronecker product between two matrices.

Estimation for correlation between longitudinal measurements, α

$$\begin{aligned}\frac{\partial F_i^{-1}(\alpha)}{\partial \alpha} &= \frac{\partial R_i^{-1}(\alpha)}{\partial \alpha} \otimes C_i^{-1} \\ &= \frac{1}{(1-\alpha^2)^2} \begin{pmatrix} 2\alpha C_i^{-1} & -(1+\alpha^2)C_i^{-1} & 0 & 0 \\ -(1+\alpha^2)C_i^{-1} & 4\alpha C_i^{-1} & -(1+\alpha^2)C_i^{-1} & 0 \\ 0 & -(1+\alpha^2)C_i^{-1} & 4\alpha C_i^{-1} & -(1+\alpha^2)C_i^{-1} \\ 0 & 0 & -(1+\alpha^2)C_i^{-1} & 2\alpha C_i^{-1} \end{pmatrix}\end{aligned}$$

With AR1 working correlation

```
alpha_stg1_ar1 <- function(mdat, Z, Qinv){
  Fa <- Fb <- 0
  for (i in mdat$cluster_id) { # for each pair
    S1_j <- S2_j <- S1_ja <- S1_jb <- 0

    t_i1 <- nlevels(as.factor(mdat[mdat$cluster_id == i & mdat$cluster.var==1,]$visit))
    t_i2 <- nlevels(as.factor(mdat[mdat$cluster_id == i & mdat$cluster.var==2,]$visit))
    t_ij <- min(c(t_i1, t_i2))
    Z_i1 <- Z[mdat$cluster_id == i & mdat$cluster.var == 1]
    matZ_i1 <- matrix(Z_i1, nrow = t_ij)
    Z_i2 <- Z[mdat$cluster_id == i & mdat$cluster.var == 2]
    matZ_i2 <- matrix(Z_i2, nrow = t_ij)

    if (t_ij > 1) {
      for (k in 1:(t_ij-1)) {
        matZ1 <- matrix(c(matZ_i1[k],matZ_i2[k]),nrow=2)
        matZ2 <- matrix(c(matZ_i1[k+1],matZ_i2[k+1]),nrow=2)
        S2_j <- S2_j + t(matZ1) %*% Qinv %*% matZ2
      }
```

```
      if (t_ij ==2){
        for(k in 1:t_ij){
          matZ <- matrix(c(matZ_i1[k],matZ_i2[k]),nrow=2)
          S1_j <- S1_j + t(matZ) %*% Qinv %*% matZ
        }
      }else{
        for(k in 1:t_ij){
          matZ <- matrix(c(matZ_i1[k],matZ_i2[k]),nrow=2)
          S1_ja <- S1_ja + t(matZ) %*% Qinv %*% matZ
        }
        for(k in 2:(t_ij-1)){
          matZ <- matrix(c(matZ_i1[k],matZ_i2[k]),nrow=2)
          S1_jb <- S1_jb + t(matZ) %*% Qinv %*% matZ
        }
        S1_j <- S1_ja + S1_jb
      }
    }

    Fa <- Fa + S1_j
    Fb <- Fb + S2_j
  }
  ### stage 1 estimate of alpha
  alpha0 <- ( Fa - sqrt( ( Fa - 2 * Fb ) * ( Fa + 2 * Fb ) ) ) / ( 2 * Fb )
  return(alpha0)
}
```

```
alpha_stg2_ar1 <- function(alpha0){
  alpha <- as.numeric( 2 * alpha0 / ( 1 + alpha0 ^ 2 ) )
  return(alpha)
}
```

With Exchangeable working correlation

```
estalpha1_exch <- function(mdat, Z, Qinv){
  match.call()
  alphafun <- function(alpha){
    GG1 <- GG2 <- 0
    for (i in unique(mdat$cluster_id)){
      GG1j <- GG2j <- 0

      t_i1 <- nlevels(as.factor(mdat[mdat$cluster_id == i & mdat$cluster.var==1,]$visit))
      t_i2 <- nlevels(as.factor(mdat[mdat$cluster_id == i & mdat$cluster.var==2,]$visit))
      t_ij <- max(c(t_i1, t_i2))
      Z_i1 <- Z[mdat$cluster_id == i & mdat$cluster.var == 1]
      if (t_i1 < t_ij) {Z_i1 <- c(Z_i1, rep(0, t_ij - t_i1))}
      matZ_i1 <- matrix(Z_i1, nrow = t_ij)
      Z_i2 <- Z[mdat$cluster_id == i & mdat$cluster.var == 2]
      if (t_i2 < t_ij) {Z_i2 <- c(Z_i2, rep(0, t_ij - t_i2))}
      matZ_i2 <- matrix(Z_i2, nrow = t_ij)

      #if(t_ij > 1){
```

```
      g1 <- vector()
      for(t in 1:t_ij){
        matZ <- matrix(c(matZ_i1[t],matZ_i2[t]),nrow=2)
        g1[t] <- t(matZ) %*% Qinv %*% matZ
      }
      G1 <- sum(g1)

      g2 <- vector()
      G2 <- 0 #
      if(t_ij > 1){ #
        for(t in 1:(t_ij - 1)){
          for(tt in (t+1):t_ij){
            matZ1 <- matrix(c(matZ_i1[t],matZ_i2[t]),nrow=2)
            matZ2 <- matrix(c(matZ_i1[tt],matZ_i2[tt]),nrow=2)
            g2 <- c(g2, t(matZ1) %*% Qinv %*% matZ2)
          }
        }
        G2 <- sum(g2)

      }
      denom <- ( 1 + ( t_ij - 1 ) * alpha ) ^ 2
      num1 <- alpha ^ 2 * ( t_ij - 1 ) * ( t_ij - 2 ) + 2 * alpha * ( t_ij - 1 )
      num2 <- ( 1 + alpha ^ 2 * ( t_ij - 1 ) )

      GG1j <- GG1j + ( G1 * num1 ) / denom
      GG2j <- GG2j + ( G2 * num2 ) / denom

      GG1 <- GG1 + GG1j
      GG2 <- GG2 + GG2j
    }
    GG1 - 2 * GG2
  }
  ### stage 1 estimate of alpha
  alpha0 <- uniroot(alphafun, c(0,1), tol = 1e-10, extendInt = "yes")$root
  return(alpha0)
}

estalpha2_exch <- function(alpha0, mdat){
  match.call()
  alphapart1 <- alphapart2 <- 0

  for (i in mdat$cluster_id){
    alphapart1j <- alphapart2j <- 0

      t_ij <- 5 #nlevels(as.factor(mdat[mdat$cluster_id == i & mdat$cluster.var == j,]$visit))
      if(t_ij > 1){
        alphapart1num <- alpha0 * ( t_ij - 1 )* ( alpha0 * (t_ij - 2) + 2 )
        alphapart2num <- ( t_ij - 1 ) * ( 1 + alpha0 ^ 2 * (t_ij - 1) )
        alphaden <- ( 1 + alpha0 * ( t_ij - 1 ) ) ^ 2

        alphapart1j <- alphapart1j + alphapart1num / alphaden
        alphapart2j <- alphapart2j + alphapart2num / alphaden
      }
```

```
    alphapart1 <- alphapart1 + alphapart1j
    alphapart2 <- alphapart2 + alphapart2j
  }
  alpha <- alphapart1 / alphapart2
  return(alpha)
}
```

Estimation for correlation within matched pair, τ

Assumptions
1. $\tau_i = \tau_i'$ for all matched-pair i, i'.
2. Exchangeable working correlation with the off-diagonal element equal to τ.

```
tau_stg1 <- function(mdat, maxT, Z, time.str,alpha0){
  Fa <- Fb <- 0
  for (i in mdat$cluster_id){
    a_1 <- a_2 <- 0
    t_i1 <- nlevels(as.factor(mdat[mdat$cluster_id == i & mdat$cluster.var==1,]$visit))
    t_i2 <- nlevels(as.factor(mdat[mdat$cluster_id == i & mdat$cluster.var==2,]$visit))
    if (time.str == "ind") {Rinv1 <- solve(diag(t_i1))}
    if (time.str == "ar1") {Rinv1 <- solve(ar1_cor(alpha0, t_i1))}
    if (time.str == "exch") {Rinv1 <- solve(exch_cormat(alpha0, t_i1))}
    if (time.str == "ind") {Rinv2 <- solve(diag(t_i2))}
    if (time.str == "ar1") {Rinv2 <- solve(ar1_cor(alpha0, t_i2))}
    if (time.str == "exch") {Rinv2 <- solve(exch_cormat(alpha0, t_i2))}
    Rinv <- solve(exch_cormat(alpha0, maxT))

    Z_i1 <- Z[mdat$cluster_id == i & mdat$cluster.var == 1]
    matZ_i1 <- matrix(Z_i1, nrow = t_i1)
    Z_i2 <- Z[mdat$cluster_id == i & mdat$cluster.var == 2]
    matZ_i2 <- matrix(Z_i2, nrow = t_i2)
    a_1 <- a_1 + t(matZ_i1) %*% Rinv1 %*% matZ_i1 + t(matZ_i2) %*% Rinv2 %*% matZ_i2

    if (maxT > t_i1) {matZ_i1 <- c(matZ_i1, rep(0, maxT - t_i1))}
    if (maxT > t_i2) {matZ_i2 <- c(matZ_i2, rep(0, maxT - t_i2))}
    a_2 <- a_2 + t(matZ_i1) %*% Rinv %*% matZ_i2

    Fa <- Fa + a_1
    Fb <- Fb + a_2

  }
  ### stage 1 estimate of tau
  tau0 <- ( Fa - sqrt( ( Fa - 2 * Fb ) * ( Fa + 2 * Fb ) ) ) / ( 2 * Fb )
  return(tau0)
}
```

```
tau_stg2 <- function(tau0){
  tau <- as.numeric( 2 * tau0 / ( 1 + tau0 ^ 2 ) )
  return(tau)
}
```

Functions for working correlations

Exchangeable

```
exch_cormat <- function(rho, n) {
  # Create an nxn matrix filled with tau
  cor_matrix <- matrix(rho, n, n)
  # Set the diagonal to 1
  diag(cor_matrix) <- 1
  return(cor_matrix)
}
```

AR1

```
ar1_cor <- function(rho,n) {
  rho <- as.numeric(rho)
  exponent <- abs(matrix(1:n - 1, nrow = n, ncol = n, byrow = TRUE) -
                    (1:n - 1))
  rho^exponent
}
```

Function for working covariance matrix Σ_i for Y_i

$$\Sigma_i = A_i^{1/2}(\beta)F_i(\Gamma)A_i^{1/2}(\beta)$$

```
Sigma <- function(data,tau, alpha, time.str, time.var){
  Sigma_list <- list()
  for (i in unique(data$cluster_id)) {
    Qi <- exch_cormat(tau, 2)
    ti1 <- nlevels(as.factor(data[data$cluster_id == i & data$cluster.var==1,]$visit))
    ti2 <- nlevels(as.factor(data[data$cluster_id == i & data$cluster.var==2,]$visit))
    ni <- ti1+ti2
    if (ti1 >= ti2)
    {
      if (time.str == "ind") {Ri <- diag(ti1)}
      if (time.str == "ar1") {Ri <- ar1_cor(alpha, ti1)}
      if (time.str == "exch") {Ri <- exch_cormat(alpha, ti1)}
      Fi <- kronecker(Qi,Ri)
      Sigma_i <- Fi[1:ni, 1:ni]
      Sigma_list <- c(Sigma_list, list(Sigma_i))
    }
    else {
      if (time.str == "ind") {Ri <- diag(ti2)}
      if (time.str == "ar1") {Ri <- ar1_cor(alpha, ti2)}
      if (time.str == "exch") {Ri <- exch_cormat(alpha, ti2)}
      Fi <- kronecker(Qi,Ri)
      Sigma_i <- Fi[1:ni, 1:ni]
      Sigma_list <- c(Sigma_list, list(Sigma_i))
    }
```

```
  }
  Sigma_list
}
```

Function for estimating β.

```
beta_hat <- function(formula,data, time.var, time.str, tau, alpha) {
  X <- model.matrix(object=formula, data = data) #design matrix
  y <- as.matrix(data$root) #response variable
  Sigma_list <- list()
  Xt_Sigma_inv_X <- list()
  Xt_Sigma_inv_y <- list()
  S <- Sigma(data=data, tau=tau, alpha=alpha, time.str=time.str, time.var=time.var)
  for (i in 1:length(S)) {
    ti1 <- nlevels(as.factor(data[data$cluster_id == i & data$cluster.var==1,]$visit))
    ti2 <- nlevels(as.factor(data[data$cluster_id == i & data$cluster.var==2,]$visit))
    if (ti1 >= ti2){
      Xi <- rbind(X[data$cluster_id==i & data$cluster.var==1,],
                  X[data$cluster_id==i & data$cluster.var==2,])
      yi <- rbind(as.matrix(y[data$cluster_id==i & data$cluster.var==1,]),
                  as.matrix(y[data$cluster_id==i & data$cluster.var==2]))
    }
    else {
      Xi <- rbind(X[data$cluster_id==i & data$cluster.var==2,],
                  X[data$cluster_id==i & data$cluster.var==1,])
      yi <- rbind(as.matrix(y[data$cluster_id==i & data$cluster.var==2,]),
                  as.matrix(y[data$cluster_id==i & data$cluster.var==1]))
    }
    Sigma_inv <- solve(S[[i]])
    Xt_Sigma_inv_X_i <- t(Xi) %*% Sigma_inv %*% Xi
    Xt_Sigma_inv_X[[i]] <- Xt_Sigma_inv_X_i
    Xt_Sigma_inv_y_i <- t(Xi) %*% Sigma_inv %*% yi
    Xt_Sigma_inv_y[[i]] <- Xt_Sigma_inv_y_i
  }
  return(solve(Reduce("+", Xt_Sigma_inv_X)) %*% Reduce("+",Xt_Sigma_inv_y))
}
```

Function for sandwich estimator

```
sandwich <- function(formula,data,beta_hat,alpha, time.str){
  X <- model.matrix(object=formula, data = data)
  y <- as.matrix(data$root)
  W <- list()
  mid <- list()
  for (i in unique(data$cluster_id)) {
    Xi <- X[data$cluster_id==i,]
    yi <- y[data$cluster_id==i]
    Zi <- yi - (Xi %*% as.matrix(beta_hat))
    ti1 <- nlevels(as.factor(data[data$cluster_id == i & data$cluster.var==1,]$visit))
```

```
    ti2 <- nlevels(as.factor(data[data$cluster_id == i & data$cluster.var==2,]$visit))
    ni <- ti1 + ti2
    Ai <- diag(ni) ^ (1/2)
    if (time.str == "ind") {Ri <- diag(ni)}
    if (time.str == "ar1") {Ri <- ar1_cor(alpha, ni)}
    if (time.str == "exch") {Ri <- exch_cormat(alpha, ni)}
    Wi <- t(Xi) %*% Ai %*% solve(Ri) %*% Ai %*% Xi
    W[[i]] <- Wi
    mid_i <- t(Xi) %*% Ai %*% solve(Ri) %*% Zi %*% t(Zi) %*% solve(Ri) %*% Ai %*% Xi
    mid[[i]] <- mid_i
  }
  Wn_inv <- solve(Reduce("+", W))
  mid_n <- Reduce("+", mid)
  out <- list()
  out$vcov <- Wn_inv %*% mid_n %*% Wn_inv
  out$se <- sqrt(diag(out$vcov))
  return(out)
}
```

Function for fitting QLS

```
qls <- function(formula, data, subject_id, cluster_id , cluster.var, time.var, maxT, time.str){
  iter <- 0
  bdiff <- c(1,1,1,1)
  alpha0  <- 0.1 # initial alpha estimate

  # use independent GEE to get initial beta estimates
  init_mod <- geeglm(formula, data = data, family = binomial('logit'),
                     id = pid, waves = visit, corstr = "independence")
  #summary(init_mod)
  beta0 <- as.vector(coef(init_mod))
  Z0 <- residuals(init_mod,"pearson") #init_mod$residuals Z0[1:5][1,]

  # compute initial tau estimate
  tau0 <- tau_stg1(mdat=data, maxT=maxT, Z = Z0, time.str = time.str,alpha0=alpha0)

  while(max(abs(bdiff)) > .00000001){
    betahat <- beta_hat(formula=formula,data=data, time.var=time.var,
                        time.str=time.str, tau=tau0, alpha=alpha0)
    beta1 <- as.vector(betahat)
    if (all(!is.na(betahat))){bdiff <- beta1 - beta0} #***

    # update tau0
    Z1 <- as.matrix(data$root) -
      model.matrix(object=formula, data = data) %*% as.matrix(betahat)

    tau00 <- tau_stg1(mdat=data, maxT=maxT, Z = Z1, time.str = time.str,alpha0=alpha0)
    # update alpha0 (initial alpha0 for the next iteration)
    if (!is.na(tau00)) {tau0 <- tau00}
    #print(tau0)
```

```
    Qinv <- solve(exch_cormat(tau0, 2))
    if (time.str == "ind") {alpha0 <- 0}
    if (time.str == "ar1") {alpha0 <- alpha_stg1_ar1(mdat=data, Z=Z1, Qinv=Qinv)}
    if (time.str == "exch") {alpha0 <- estalpha1_exch(mdat=data, Z=Z1, Qinv=Qinv)}

    iter <- iter + 1
    beta0 <- beta1
    # print(paste("iter:", iter, sep = " "))
    # print(paste("alpha0:",alpha0, sep = " "))
    # print(paste("tau0:",as.numeric(tau0), sep = " "))
    # print(paste("bdiff:",max(abs(bdiff)), sep = " "))
  }

  # after converge, get stage 2 estimates
  tau2 <- tau_stg2(tau0)
  if (time.str == "ind") {alpha2 <- alpha0}
  if (time.str == "ar1") {alpha2 <- alpha_stg2_ar1(alpha0)}
  if (time.str == "exch") {alpha2 <- estalpha2_exch(alpha0, mdat = data)}

  betahat1 <- beta_hat(formula=formula,data=data, time.var=time.var,
                       time.str=time.str, tau=tau2, alpha=alpha2)
  beta <- as.vector(betahat1)
  sandwich_out <- sandwich(formula = formula, data = data, beta_hat = betahat1,
                           alpha = alpha2, time.str = time.str)
  se <- sandwich_out$se
  vcov <- sandwich_out$vcov

  fit <- list()
  fit$call <- match.call()
  fit$coefficients <- beta
    fit$se <- se
  fit$alpha <- alpha2
  fit$tau <- tau2
  fit$niter <- iter
  fit$vcov <- vcov
  fit
}
```

```
sum(is.na(matched_samples$y))
```

```
## [1] 0
```

Function for calculating 95% CI for QLS estimates

```
qls_ci <- function(model, level = 0.95) {
  # Calculate the lower and upper bounds of the confidence interval for each parameter
  lower <- model$coefficients - qnorm(1-(1-0.95)/2) * model$se
  upper <- model$coefficients + qnorm(1-(1-0.95)/2) * model$se
  results <- data.frame(lower = lower, upper = upper)
  return(results)
}
```

```r
adj_qls_ci <- function(model, N, level = 0.95) {
  # Calculate the adjusted standard errors using DF-corrected sandwich estimator
  adj_se <- sqrt(diag((N/(N-p))*model$vcov))
  # Calculate the lower and upper bounds of the confidence interval for each parameter
  lower <- model$coefficients - qnorm(1-(1-0.95)/2) * adj_se
  upper <- model$coefficients + qnorm(1-(1-0.95)/2) * adj_se
  results <- data.frame(lower = lower, upper = upper)
  return(results)
}
```

Fit real data

```r
qls_data <- matched_samples[order(matched_samples$pairID),]

# create cluster id to identify each pair
uni_pairs <- unique(qls_data$pairID)
num_pair <- length(uni_pairs)
cluster_id <- seq_len(num_pair) # cluster IDs
lookup_tbl <- data.frame(pairID = uni_pairs, cluster_id = cluster_id)
qls_data <- merge(qls_data, lookup_tbl, by = "pairID")
qls_data <- qls_data %>%
  group_by(cluster_id) %>%
  mutate(cluster.var = ifelse(bav == 0, 1, 2),
         order = row_number()) %>%
  relocate(c(cluster_id, cluster.var, order), .after = pid)

qls_ind <- qls(y ~ bav*visit, data=qls_data,
               subject_id = pid, cluster_id =  cluster_id,
               cluster.var = cluster.var,
               time.var = visit, maxT = 6, time.str = "ind")

qls_ar1 <- qls(y ~ bav*visit, data=qls_data,
               subject_id = pid, cluster_id =  cluster_id,
               cluster.var = cluster.var,
               time.var = visit, maxT = 6, time.str = "ar1")

qls_exch <- qls(y ~ bav*visit, data=qls_data,
               subject_id = qls_data$pid, cluster_id =  cluster_id,
               cluster.var = cluster.var,
               time.var = qls_data$visit, maxT = 6, time.str = "exch")
```

```r
qls_results <- tibble(
  params = c(gee_results$params, "$\\tau$"),
  ind_est = c(qls_ind$coefficients, NA, qls_ind$tau),
  ind_se = c(qls_ind$se, NA, NA),
  ind_DFse = c(DFC*sqrt(diag(qls_ind$vcov)), NA, NA),
  exch_est = c(qls_exch$coefficients, qls_exch$alpha, qls_exch$tau),
  exch_se = c(qls_exch$se, NA, NA),
  exch_DFse = c(DFC*sqrt(diag(qls_exch$vcov)), NA, NA),
  ar1_est = c(qls_ar1$coefficients, qls_ar1$alpha, qls_ar1$tau),
```

```
  ar1_se = c(qls_ar1$se, NA, NA),
  ar1_DFse = c(DFC*sqrt(diag(qls_ar1$vcov)), NA, NA))
```

	Independent			Exchangeable			AR1		
Parameter	Estimate	SE	SE-DF	Estimate	SE	SE-DF	Estimate	SE	SE-DF
GEE									
β_0 (Intercept)	-2.082	0.737	0.811	-2.050	0.718	0.790	-2.128	0.730	0.803
β_1 (BAV)	-1.085	1.130	1.243	-1.183	1.120	1.232	-1.125	1.154	1.269
β_2 (Visit)	0.096	0.206	0.227	0.058	0.217	0.238	0.099	0.203	0.224
β_3 (Visit×BAV)	0.154	0.320	0.352	0.220	0.324	0.357	0.175	0.329	0.362
α				0.122	0.111		0.169	0.169	
QLS									
β_0 (Intercept)	34.151	1.857	2.043	34.306	1.683	1.851	34.506	1.626	1.789
β_1 (BAV)	-0.081	2.272	2.499	-0.271	2.201	2.421	0.287	2.588	2.847
β_2 (Visit)	-0.163	0.499	0.549	-0.452	0.393	0.433	-0.437	0.576	0.634
β_3 (Visit×BAV)	0.139	0.622	0.684	0.565	0.574	0.632	0.300	0.663	0.729
α				0.663			0.707		
τ	0.053			0.041			0.043		