

STAT-6494 Advanced Statistical Computing with R

Homework 5

Wenjie Wang

16 March 2016

1 Exercise: A Simple Implementation of QIF.

Write a function `simpleQIF` that evaluates the QIF $Q_n(\beta)$ at regression coefficient vector β for a given response vector Y , covariate matrix X , an id vector that distinguishes the clusters, and an exchangeable given working correlation structure. For the example dataset `ohio` in package `geepack`, feed your function to an optimizer such as `optim` to solve for the parameter estimate. An example call would look like `simpleQIF(y, X, id, family)`.

1.1 Function `simpleQIF`

Qu, Lindsay, and Li (2000) proposed quadratic inference function (QIF) to improve generalized estimating equations. Here, we consider logit as link function for binary outcomes as default and only assume exchangeable correlation structure. The simple implementation of QIF, `simpleQIF` is designed to return estimates of covariate coefficients. The function body is given as follows.

```
simpleQIF <- function (y, X, id, family = binomial) {
  X <- as.matrix(X)
  y <- as.vector(y)
  famFun <- family()
  ## balanced cluster size
  idTab <- as.numeric(table(id))
  clusN <- length(idTab)
  clusSize <- unique(idTab)
  if (length(clusSize) != 1) stop("Not balanced cluster size.")
  ## define matrix m_0 and matrix m_1
  ## for exchangeable structure
  mat0 <- diag(clusSize)
  mat1 <- matrix(1, clusSize, clusSize) - mat0
  matMList <- list(mat0, mat1)
  ## initialize beta as starting point
  fit0 <- glm.fit(X, y, family = famFun)
  beta0 <- fit0$coef
  ## optim step
  res <- optim(beta0, fn = qNbeta, gr = NULL,
              clusSize = clusSize, clusN = clusN, y = y, X = X,
              matMList = matMList, famFun = famFun)
  ## simply output coef estimates
  res$par
}

### internal functions
## computes g_ri(beta) for cluster i and matrix r
```

```

g_riBeta <- function (matM, betaVec, yi, Xi, famFun) {
  eta <- as.vector(Xi %*% betaVec, mode = "numeric")
  muiVec <- famFun$linkinv(eta)
  matDi <- famFun$mu.eta(muiVec) * Xi # element-wise
  sqrtInvAi <- diag(1 / sqrt(famFun$variance(muiVec)))
  crossprod(matDi, sqrtInvAi) %*% (matM %*% (sqrtInvAi %*% (yi - muiVec)))
}

## computes g_i(beta) for cluster i
g_iBeta <- function (betaVec, matMList, yi, Xi, famFun) {
  out <- lapply(matMList, g_riBeta, betaVec = betaVec, yi = yi,
               Xi = Xi, famFun = famFun)
  do.call("c", out)
}

## computes QIF: Q_N(beta)
qNbeta <- function (betaVec, clusSize, clusN, y, X, matMList, famFun) {
  nMat <- length(matMList)
  nBeta <- length(betaVec)
  gNbeta <- matrix(NA, nrow = clusN, ncol = nMat * nBeta)
  indBase <- seq(1 - clusSize, 0)
  for (i in seq_len(clusN)) {
    ind <- indBase + i * clusSize
    Xi <- X[ind, ]
    yi <- y[ind]
    gNbeta[i, ] <- g_iBeta(betaVec, matMList, yi, Xi, famFun)
  }
  gBarN <- colMeans(gNbeta)
  cNbeta <- var(gNbeta)
  clusN * crossprod(gBarN, MASS::ginv(cNbeta) %*% gBarN)
}

```

1.2 Comparison with geepack on a Sample Dataset

We compare the estimates of covariate coefficients produced by **geese** in **geepack** to verify our implementation of QIF on a sample dataset, *ohio*.

From the comparison, we may find that the estimates by **simpleQIF** and **geese** are quite close to each other. So we have confidence that the implementation is correct.

```

require(geepack)
data(ohio)
y <- ohio$resp
X <- cbind("(Intercept)" = 1, ohio[, 3:4])
id <- ohio$id
simpleQIF(y, X, id) # estimates from our implementation

```

```

(Intercept)      age      smoke
-1.8844759  -0.1189905   0.2637976

```

```

## compare with geepack
fitGee <- geese(resp ~ age + smoke, id = id, data = ohio,

```

```
family = binomial, corstr = "exch", scale.fix = TRUE)
fitGee$beta # estimates from geepack
```

(Intercept)	age	smoke
-1.8804288	-0.1133850	0.2650833

Reference

Qu, Annie, Bruce G Lindsay, and Bing Li. 2000. "Improving Generalised Estimating Equations Using Quadratic Inference Functions." *Biometrika* 87 (4). Biometrika Trust: 823–36.