```
This file contains an R program, which serves as supplemental material
for the article "Robust methods for moderation analysis with a two-level
regression model" in the journal Multivariate Behavioral Research.

Authors: Miao Yang and Ke-Hai Yuan

##=======================================##
##  A simulated data set with 500 cases  ##
##=======================================##
set.seed(1113)
# generate the dependent variable from Student's t distribution with df=5
DV <- rt(500,5)
# generate the independent variable from the standard normal distribution
IV <- rnorm(500)
# generate the moderator from a uniform distribution U(1,10)
Moderator <- runif(500,1,10)
data <- cbind(DV,IV,Moderator)
write.table(data,'simdata.txt',row.names=F)

##=========================================================================##
##  Functions for moderation analysis with two-level regression model
##  (Do NOT change codes in the functions!)
##=========================================================================##
require(MASS)

# Function for moderation analysis using normal-distribution-based ML
NML<-function(y,x,z,data=NULL){
  arguments <- as.list(match.call())
  param.names <- arguments[2:4]
  if(!is.null(data)){
    x <- eval(arguments$x,data)
    z <- eval(arguments$z,data)
    y <- eval(arguments$y,data)
  }

  n <- length(y)
  h <- cbind(x^2,2*x,rep(1,n))
  C <- cbind(rep(1,n),z,x,x*z)

  LS.MMR<-lm(y~x*z)
  # Using LS estimates as starting values for regression coefficients
  r00<-LS.MMR$coefficients[1]
  r01<-LS.MMR$coefficients[3]
  r10<-LS.MMR$coefficients[2]
  r11<-LS.MMR$coefficients[4]
  gamma.start <- as.matrix(c(r00,r01,r10,r11))
  # Starting values for variance estimates
  sigma.start <- c(1,0,1)

  for (iter in 1:500) {

    tau2<-h%*%sigma.start

    A<-diag(as.vector(1/tau2))
    B<-diag(as.vector(1/tau2^2))

    gamma<-solve(t(C)%*%A%*%C)%*%(t(C)%*%A%*%y)
    delta2<-(y-C%*%gamma)^2
    sigma<-solve(t(h)%*%B%*%h)%*%(t(h)%*%B%*%delta2)
```

```
    diff<-sum(abs(gamma-gamma.start))+sum(abs(sigma-sigma.start))

    gamma.start<-gamma
    sigma.start<-sigma

    if (diff<.0001) {
      conv<-1
      break
    }
  }

  tau2<-h%*%sigma
  delta2<-(y-C%*%gamma)^2
  delta<-y-C%*%gamma

  #BIC
  k <-length(gamma)+length(sigma)
  BIC <- n*log(2*pi) + sum(log(tau2)) + sum((delta2)/tau2) + k*log(n)

  #Sandwich SE
  U11<-t(C)%*%diag(as.vector(1/tau2))%*%C;
  U12<-t(C)%*%diag(as.vector(delta/tau2^2))%*%h
  U21<-t(U12);
  U22<-t(h)%*%diag(as.vector((2*delta2-tau2)/(2*tau2^3)))%*%h

  V11<-t(C)%*%diag(as.vector(delta2/tau2^2))%*%C;
  V12<-t(C)%*%diag(as.vector((delta^3-delta*tau2)/(2*tau2^3)))%*%h;
  V21<-t(V12);
  V22<-t(h)%*%diag(as.vector((delta2-tau2)^2/(4*tau2^4)))%*%h

  U<-rbind(cbind(U11,U12),cbind(U21,U22))
  V<-rbind(cbind(V11,V12),cbind(V21,V22))
  Cov<-solve(U)%*%V%*%solve(U)
  se.theta<-sqrt(diag(Cov))

  theta<-c(gamma,sigma)
  zvalue<-theta/se.theta
  p<-2*pnorm(-abs(zvalue))
  R.square<-(gamma[4])^2*var(z)/((gamma[4])^2*var(z)+sigma[1])

  out<-NULL

  Parameter <- cbind(theta,se.theta,zvalue,p)
  rownames(Parameter) <-
c("Intercept",param.names[3:2],paste(param.names[2],param.names[3],sep="*
"),'Var1','Cov01','Var(0+e)')
  colnames(Parameter) <- c("Estimate","SE_sw","z","Pr(>|z|)")
  out <- list(Parameter=Parameter,BIC=BIC,R.square=R.square)
  return(out)
}

# Function for moderation analysis using Student's t distribution
TML<-function(y,x,z,m=5,data=NULL){
  arguments <- as.list(match.call())
  param.names <- arguments[2:4]
  if(!is.null(data)){
    x <- eval(arguments$x,data)
    z <- eval(arguments$z,data)
    y <- eval(arguments$y,data)
  }
```

```
  n <- length(y)
  h <- cbind(x^2,2*x,rep(1,n))
  C <- cbind(rep(1,n),z,x,x*z)

  LS.MMR<-lm(y~x*z)
  r00<-LS.MMR$coefficients[1]
  r01<-LS.MMR$coefficients[3]
  r10<-LS.MMR$coefficients[2]
  r11<-LS.MMR$coefficients[4]
  gamma.start <- as.matrix(c(r00,r01,r10,r11))
  sigma.start <- c(1,0,1)

  for (iter in 1:500) {

    delta<-y-C%*%gamma.start
    tau2<-h%*%sigma.start*(m-2)/m

    di2<-delta^2/tau2
    wi<-(m+1)/(m+di2)

    d<-as.matrix(di2)
    A<-diag(as.vector(wi/tau2))
    B<-diag(as.vector(1/tau2^2))

    gamma<-solve(t(C)%*%A%*%C)%*%(t(C)%*%A%*%y)
    sigma<-solve(t(h)%*%B%*%h)%*%(t(h)%*%A%*%d)*m/(m-2)

    diff<-sum(abs(gamma-gamma.start))+sum(abs(sigma-sigma.start))

    gamma.start<-gamma
    sigma.start<-sigma

    if (diff<.0001) {
      conv<-1
      break
    }
  }

  delta<-y-C%*%gamma.start;
  delta2<-(y-C%*%gamma.start)^2;
  tau2<-h%*%sigma.start*(m-2)/m;
  d<-as.matrix(delta2/tau2)
  di2<-delta^2/tau2
  wi<-(m+1)/(m+di2)

  #BIC
  k <-length(gamma)+length(sigma)
  BIC <- 2*n*log(sqrt(m*pi)*gamma(m/2)/gamma(m/2+0.5)) + sum(log(tau2)) +
(m+1)*sum(log(1+d/m)) + k*log(n)

  #Sandwich SE
  U11.part<-diag(as.vector((m-d)*wi/((d+m)*tau2))) #d is di^2
  U12.part<-diag(as.vector((m-2)*wi*delta/((d+m)*tau2^2)))
  U21.part<-U12.part
  U22.part<-diag(as.vector(0.5*(1-2/m)^2*((d+2*m)/(d+m)*wi*d-1)/tau2^2))
  V11.part<-diag(as.vector(wi^2*d/tau2))
  V12.part<-diag(as.vector((m-2)*(wi*d-1)*wi*delta/(2*m*tau2^2)))
  V21.part<-V12.part
  V22.part<-diag(as.vector((m-2)^2*(wi*d-1)^2/(4*m^2*tau2^2)))
```

```
  U11<-t(C)%*%U11.part%*%C;
  U12<-t(C)%*%U12.part%*%h;
  U21<-t(U12);
  U22<-t(h)%*%U22.part%*%h
  V11<-t(C)%*%V11.part%*%C;
  V12<-t(C)%*%V12.part%*%h;
  V21<-t(V12);
  V22<-t(h)%*%V22.part%*%h

  U<-rbind(cbind(U11,U12),cbind(U21,U22))
  V<-rbind(cbind(V11,V12),cbind(V21,V22))
  Cov<-solve(U)%*%V%*%solve(U)
  se.theta<-sqrt(diag(Cov))

  theta<-c(gamma,sigma)
  zvalue<-theta/se.theta
  p<-2*pnorm(-abs(zvalue))
  R.square<-(gamma[4])^2*var(z)/((gamma[4])^2*var(z)+sigma[1])

  out<-NULL

  Parameter <- cbind(theta,se.theta,zvalue,p)
  rownames(Parameter) <-
c("Intercept",param.names[3:2],paste(param.names[2],param.names[3],sep="*
"),'Var1','Cov01','Var(0+e)')
  colnames(Parameter) <- c("Estimate","SE_sw","z","Pr(>|z|)")
  out <- list(Parameter=Parameter,BIC=BIC,R.square=R.square)
  return(out)
}

# Function for moderation analysis using Huber-type weights
Huber<-function(y,x,z,alpha=0.05,data=NULL){
  arguments <- as.list(match.call())
  param.names <- arguments[2:4]
  if(!is.null(data)){
    x <- eval(arguments$x,data)
    z <- eval(arguments$z,data)
    y <- eval(arguments$y,data)
  }

  n <- length(y)
  h <- cbind(x^2,2*x,rep(1,n))
  C <- cbind(rep(1,n),z,x,x*z)

  LS.MMR<-lm(y~x*z)
  r00<-LS.MMR$coefficients[1]
  r01<-LS.MMR$coefficients[3]
  r10<-LS.MMR$coefficients[2]
  r11<-LS.MMR$coefficients[4]
  gamma.start <- as.matrix(c(r00,r01,r10,r11))
  sigma.start <- c(1,0,1)

  H2<-qchisq(1-alpha,1)
  H<-sqrt(H2)
  kappa=pchisq(H2,3)+H2*alpha
  q=(H^2/kappa-1)/2

  for (iter in 1:500) {

    delta<-y-C%*%gamma.start
```

```
    tau2<-h%*%sigma.start
    tau2[tau2<=0]<-0.0001

    di<-delta/sqrt(tau2)
    wi<- ifelse(abs(di)<H,1,H/abs(di))
    wi2 <- (wi^2)/kappa

    d<-as.matrix(di^2)
    A<-diag(as.vector(wi/tau2))
    A2 <- diag(as.vector(wi2/tau2))
    B<-diag(as.vector(1/tau2^2))

    gamma<-solve(t(C)%*%A%*%C)%*%(t(C)%*%A%*%y)
    sigma<-solve(t(h)%*%B%*%h)%*%(t(h)%*%A2%*%d)

    diff<-sum(abs(gamma-gamma.start))+sum(abs(sigma-sigma.start))

    gamma.start<-gamma
    sigma.start<-sigma

    if (diff<.0001) {
      conv<-1
      break
    }
  }

  tau2<-h%*%sigma;
  delta2<-(y-C%*%gamma)^2
  delta<-y-C%*%gamma
  di<-delta/sqrt(tau2)
  wi<- ifelse(abs(di)<H,1,H/abs(di))
  wi2 <- (wi^2)/kappa

  tau<-sqrt(tau2)
  di<-delta/tau

  #Sandwich SE
  U11.part<-ifelse(abs(di)<=H, 1/tau2, 0)
  U12.part<-ifelse(abs(di)<=H, delta/tau2^2, H*sign(di)/(2*tau^3))
  U21.part<-ifelse(abs(di)<=H, delta/kappa/tau2^2, 0)
  U22.part<-ifelse(abs(di)<=H, delta2/kappa/tau2^3-1/(2*tau2^2),
q/(tau2^2))

  V11.part<-ifelse(abs(di)<=H, delta2/tau2^2,H^2/tau2)
  V12.part<-ifelse(abs(di)<=H, (delta^3/kappa-
delta*tau2)/(2*tau2^3),q*H*sign(di)/(tau^3))
  V22.part<-ifelse(abs(di)<=H, (delta2/kappa-
tau2)^2/(4*tau2^4),(q^2)/(tau2^2))

  U11<-t(C)%*%diag(as.vector(U11.part))%*%C;
  U12<-t(C)%*%diag(as.vector(U12.part))%*%h
  U21<-t(h)%*%diag(as.vector(U21.part))%*%C;
  U22<-t(h)%*%diag(as.vector(U22.part))%*%h

  V11<-t(C)%*%diag(as.vector(V11.part))%*%C
  V12<-t(C)%*%diag(as.vector(V12.part))%*%h
  V21<-t(V12)
  V22<-t(h)%*%diag(as.vector(V22.part))%*%h

  U<-rbind(cbind(U11,U12),cbind(U21,U22))
```

```
  V<-rbind(cbind(V11,V12),cbind(V21,V22))
  Cov<-solve(U)%*%V%*%solve(t(U))
  se.theta<-sqrt(diag(Cov))

  theta<-c(gamma,sigma)
  zvalue<-theta/se.theta
  p<-2*pnorm(-abs(zvalue))
  R.square<-(gamma[4])^2*var(z)/((gamma[4])^2*var(z)+sigma[1])

  out<-NULL

  Parameter <- cbind(theta,se.theta,zvalue,p)
  rownames(Parameter) <-
c("Intercept",param.names[3:2],paste(param.names[2],param.names[3],sep="*
"),'Var1','Cov01','Var(0+e)')
  colnames(Parameter) <- c("Estimate","SE_sw","z","Pr(>|z|)")
  out <- list(Parameter=Parameter,BIC=NA,R.square=R.square)
  return(out)
}

##=======================================##
##  Code that needs user's modification   ##
##=======================================##
setwd("c:/moderation")
sim <- read.table('simdata.txt',header=T)
NML(data=sim,y=DV,x=IV,z=Moderator)
TML(data=sim,y=DV,x=IV,z=Moderator,m=5)
Huber(data=sim,y=DV,x=IV,z=Moderator,alpha=.05)
```