DML-IVQR: L1-QR approximation via the Huber function (hqreg)

Enviroment required

- **R version 4.0.4**
- **quantreg__5.83**
- **hdm__0.3.1**
- **hqreg__1.4**
- **mvtnorm__1.1-1**
- **doSNOW__1.0.19**
- **ggplot2**

1.1 IVQR as GMM with residualing Z on x

- **Function Input**
  - y ====> Outcome variable
  - D ====> Treatment varaible
  - X ====> Control variable
  - Z ====> Insturmental variable
  - grid ===> Grid search interval
  - tau ===> Quantile index

  - core ===> Parallel core
- **Function Output**
  - result[1] ====> Min of the GMM function in the grid (Point estimation)
  - result[2] ====> GMM with each grid

```
gmm<-function(y,D,X,Z,tau,grid,core){
    cl <- makeCluster(core)
    registerDoSNOW(cl)
    A=foreach(i = grid, .combine = "rbind") %dopar%{
        library(quantreg)
        beta<- rq(y-i*D ~ X, tau = tau)
        beta=matrix(beta$coefficients,nrow = 1)
        e=y-i*D-cbind(1,X)%*%t(beta)
        hh=sd(e)*(4/3/length(e))^(1/5)
        distribition=akj(e,z=e,h = hh)$dens
        distribition=diag(distribition)
        psi=matrix(0,nrow = length(Z[1,]),ncol=length(Z[,1]))
        for (j in 1:length(Z[1,])) {
            delta=lm(distribition%*%Z[,j] ~ distribition%*%X)
            delta=matrix(delta$coefficients,ncol=1)
            delta=Z[,j]-cbind(1,X)%*%delta
            psi[j,]=t(delta)
            j=j+1
```

```
        }
        indicator=ifelse(e<=0,1,0)
        g=(psi%*%(tau-indicator))
        invsigma=(solve(psi%*%diag(diag((tau-indicator)%*%t(tau-indicator)))%*%t(psi)))
        gmm=(t(g)%*%invsigma%*%g)
        return(gmm)
    }
    stopCluster(cl)
    I=which.min(A)
    param1=grid[I]
    result=list(param1,A)
    return(result)
}
```

1.2 Estimating DML-IVQR (cross-validation-based λ)

We utilize the algorithm developed by Yi and Huang (2017) using the Huber loss function to approximate the quantile loss function

- **Function Input**
  - y ====> Outcome variable
  - D ====> Treatment varaible
  - X ====> Control variable
  - Z ====> Insturmental variable
  - grid ===> Grid search interval
  - tau ===> Quantile index
  - core ===> Parallel core
  - cv_fold ==> L1 norm CV fold (default=5)
  - penalty ==> L1 norm CV penalty level (default=seq(0,20,length=11))
- **Function Output**
  - result[1] ====> Min of the GMM function in the grid (Point estimation)
  - result[2] ====> GMM with each grid

```
DML_IVQR<-function(y,D,X,Z,tau,grid,core,cv_fold=5,penalty=seq(0.1,0.001,length.out = 50))
{
  cl <- makeCluster(core)
  registerDoSNOW(cl)
  A=foreach(i = grid, .combine = "rbind") %dopar%{
    library(hqreg)
    library(quantreg)
    library(hdm)
    lasso=cv.hqreg(X,y-i*D,method=c("quantile"),tau=tau,FUN = c("hqreg")
                   ,nlambda = 50,lambda=penalty,nfolds = cv_fold,
                   type.measure = c("mae"),seed = 2021)
    cv.beta=as.matrix(lasso$fit$beta)
    kfold=which(lasso$lambda==lasso$lambda.min, arr.ind=T )
    kfold.beta=cv.beta[,kfold]
    beta=matrix(kfold.beta,nrow = 1)
    e=y-i*D-cbind(1,X)%*%t(beta)
    hh=sd(e)*(4/3/length(e))^(1/5)
    distribition=akj(e,z=e,h = hh)$dens
```

```
    distribition=diag(distribition)
    psi=matrix(0,nrow = length(Z[1,]),ncol=length(Z[,1]))
    for (j in 1:length(Z[1,])) {
      delta=rlasso(distribition%*%Z[,j] ~ distribition%*%X, post = FALSE)
      delta=matrix(delta$coefficients,ncol=1)
      delta=Z[,j]-cbind(1,X)%*%delta
      psi[j,]=t(delta)
      j=j+1
    }
    indicator=ifelse(e<=0,1,0)
    g=(psi%*%(tau-indicator))
    invsigma=(solve(psi%*%diag(diag((tau-indicator)%*%t(tau-indicator)))%*%t(psi)))
    gmm=(t(g)%*%invsigma%*%g)
    return(gmm)
  }
  stopCluster(cl)
  I=which.min(A)
  param1=grid[I]
  result=list(param1,A)
  return(result)
}
```

2.1 Data Generating Process. cf. Chen, Huang, and Tien (2021, Section 3)

```
library(mvtnorm)
set.seed(2021)
sample_size=1000
n = sample_size
p = 100
s=7
sigma <- matrix(c(1,0.3,0.3,1), ncol=2)
epsilon<-rmvnorm(n=n, mean=c(0,0), sigma=sigma)
x= matrix(rnorm(n * p), ncol = p)
X=matrix(pnorm(x),ncol = p)
z<-matrix(cbind(rnorm(n,0,1),rnorm(n,0,1)),ncol = 2)
d<-z[,1]+z[,2]+epsilon[,2]
D<-pnorm(d)
Z1<-z[,1]+rnorm(n,0,1)+X[,2]+X[,3]+X[,4]
Z2<-z[,2]+rnorm(n,0,1)+X[,7]+X[,8]+X[,9]+X[,10]
Z<-matrix(cbind(Z1,Z2),nrow = n)
b = matrix(c(rep(5, s), rep(0, p - s)))
X1=X[,c(1:10)]
y=1+D+X%*%b+(epsilon[,1]*D)
```

2.2 Estimation

```
library(doSNOW)
```

```
## Loading required package: foreach
```

```
## Loading required package: iterators
```

```
## Loading required package: snow
```

```
grid=seq(-1,3,length=41)
Oracle_qr50=gmm(y,D,X1,Z,tau=0.5,grid,core=4)
DML_qr50=DML_IVQR(y,D,X,Z,tau=0.5,grid,core=4)
summary(Oracle_qr50)
```

```
##      Length Class  Mode
## [1,]  1     -none- numeric
## [2,] 41     -none- numeric
```

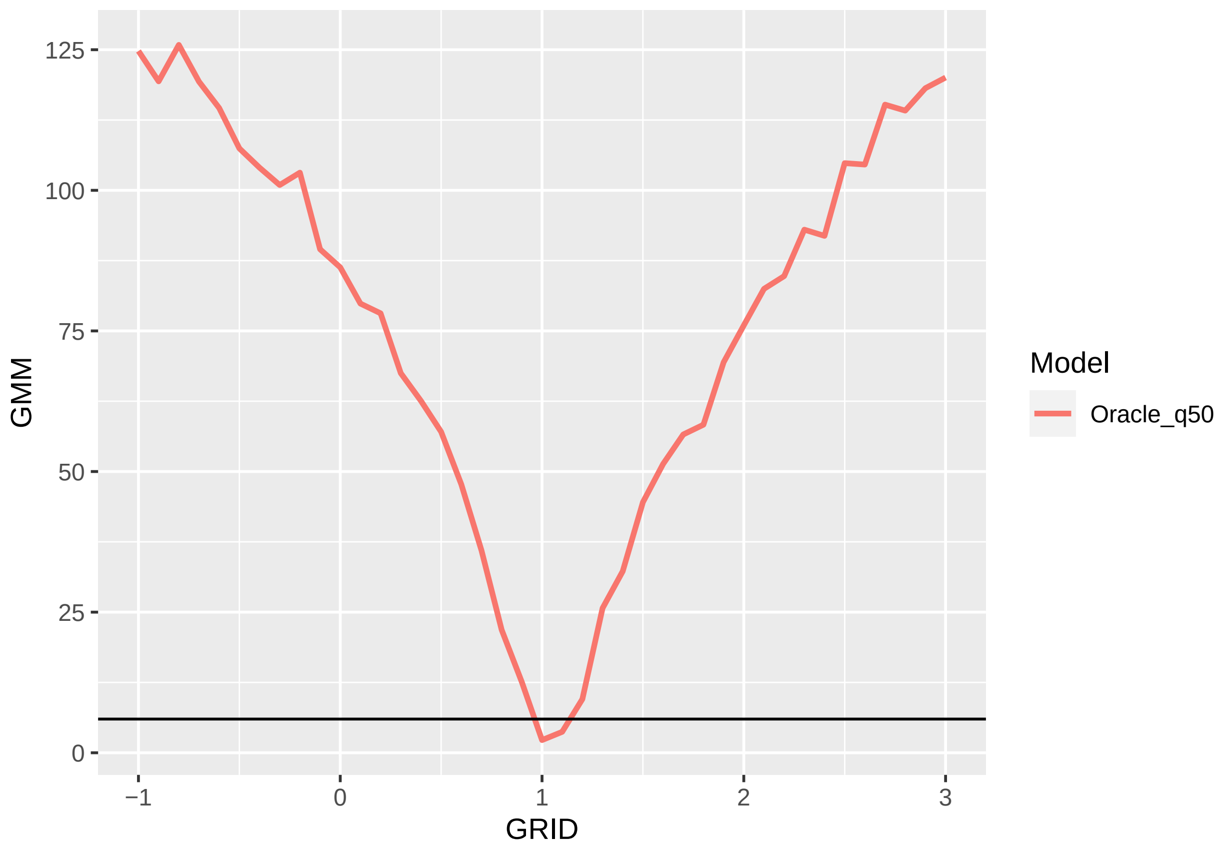
```
summary(DML_qr50)
```

```
##      Length Class  Mode
## [1,]  1     -none- numeric
## [2,] 41     -none- numeric
```

2.3 Weak-Instrument Robust Inference with Oracle Model

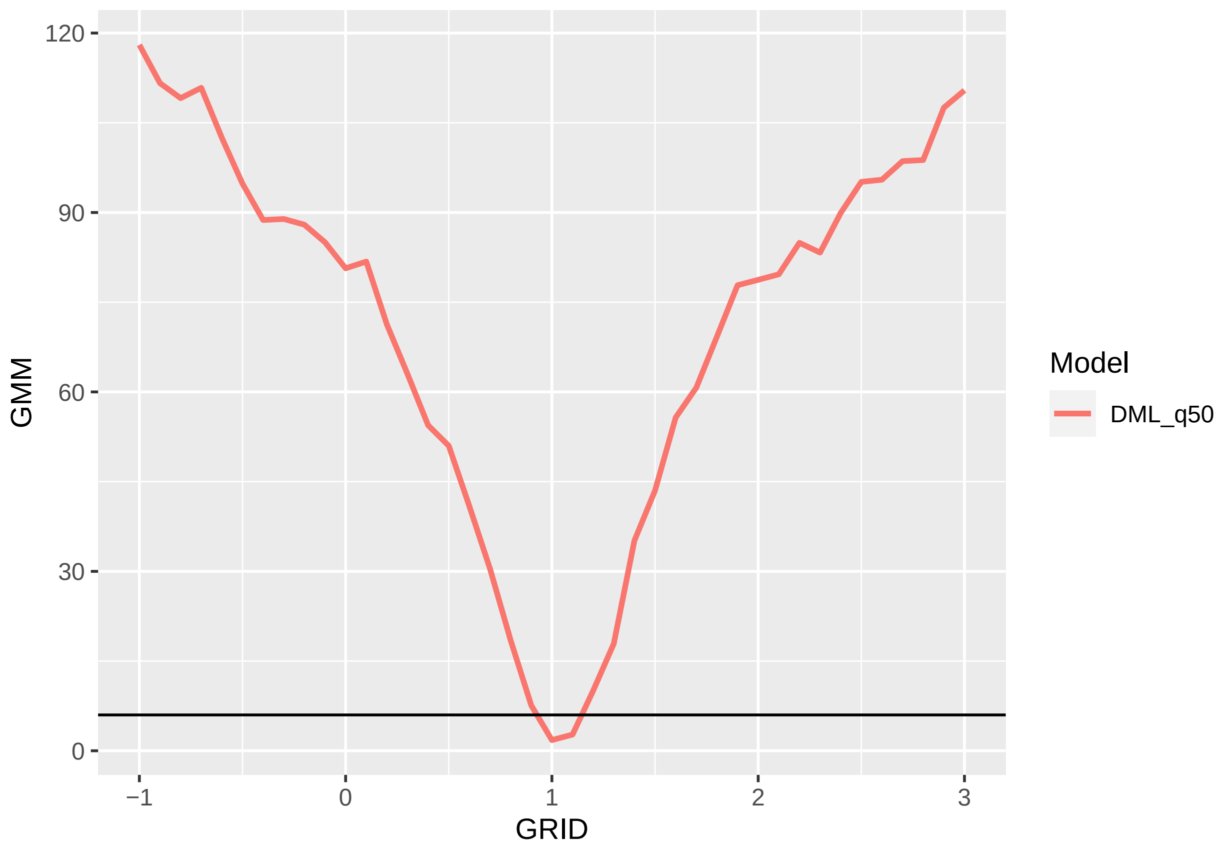
```
library(ggplot2)

GMM=unlist(Oracle_qr50[2])
GRID=seq(-1,3,length=length(grid))
Model=rep("Oracle_q50",length(grid))
tgg=data.frame(Model,GRID,GMM)
qr50=ggplot(tgg, aes(x=GRID, y=GMM, colour=Model,group=Model)) + geom_line(size=1) +
  xlim(-1,3) +  geom_hline(aes(yintercept =  5.9915))
qr50
```



2.4 Weak-Instrument Robust Inference with DML-IVQR

```
GMM=unlist(DML_qr50[2])
GRID=seq(-1,3,length=length(grid))
Model=rep("DML_q50",length(grid))
tgg=data.frame(Model,GRID,GMM)
qr50=ggplot(tgg, aes(x=GRID, y=GMM, colour=Model,group=Model)) + geom_line(size=1) +
  xlim(-1,3) +  geom_hline(aes(yintercept =  5.9915))
qr50
```



120
90
60
30
0
GMM
−1
0
1
2
3
GRID
Model
DML_q50