DML-IVQR: L1-QR penalty based on Belloni and Chernozhukov (2011, Annals of Statistics) or cross-validation (quantreg)

Enviroment required

- **R version 4.0.4**
- **quantreg_5.83**
- **hdm_0.3.1**
- **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

The penalty level λ in the L1-QR is either calculated by the theoretical formula developed in Belloni and Chernozhukov (2011) or by k-fold cross-validation.

- **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 =====> True is cross validation , False is Belloni and Chernozhukov (2011) (default=FALSE)
  - 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_alpha,core,CV=FALSE,CV_fold=5,
                   penalty=seq(0,20,length=11)){
  cl <- makeCluster(core)
  registerDoSNOW(cl)
  A=foreach(i = grid_alpha, .combine = "rbind") %dopar%{
    library(quantreg)
    library(hdm)
    norm2n<- function(z){  sqrt(mean(z^2)) }
    cv_qr_penalty<-function(y,X,tau=tau,grid,kfold){
      valid_fold=rep(0,kfold)
      sample_size=dim(y)[1]
      index=sample(rep(1:sample_size))
      grid_mae=matrix(0,ncol=length(grid),nrow=kfold)
      for (cf in 1:kfold) {
        out_index=index[(((cf-1)/kfold)*sample_size+1):(cf/kfold*sample_size)]
        y_out = matrix(y[out_index]);X_out = X[out_index,]
```

```
      start=0
      for (out in 1:kfold) {
        if(out==cf){
          next
        }
        in_index=index[(((out-1)/kfold)*sample_size+1):(out/kfold*sample_size)]
        if(start==0){
          y_in = matrix(y[in_index]);X_in = X[in_index,]
          start=1
        }
        else{
          y_in = rbind(y_in,matrix(y[in_index]));X_in = rbind(X_in,X[in_index,])
        }
      }
      for (i in 1:length(grid)) {
        fit=rq(y_in ~ X_in,tau=tau, method="lasso",lambda = grid[i])
        beta=matrix(fit$coefficients,ncol = 1)
        e1=sum(abs(y_out-cbind(1,X_out)%*%beta))/dim(y)[1]
        grid_mae[cf,i]=e1
      }
    }
    grid_mae=colSums (grid_mae, na.rm = FALSE, dims = 1)/kfold
    I=which.min(grid_mae)
    return(grid[I])
  }
  lambda.BC<- function(X, R = 1000, tau = 0.5, c = 2, alpha = .1){
    n <- nrow(X)
    sigs <- apply(X,2,norm2n)
    U <- matrix(runif(n * R),n)
    R <- (t(X) %*% (tau - (U < tau)))/(sigs*sqrt(tau*(1-tau)))
    r <- apply(abs(R),2,max)
    c * quantile(r, 1 - alpha) * sqrt(tau*(1-tau))*c(1,sigs)
  }
  if(CV == FALSE){
    lasso=rq(y-i*D ~ X,tau=tau, method="lasso",
             lambda = lambda.BC(X,tau=tau,c=2, alpha=0.1))
  }
  else{
    lasso=rq(y-i*D ~ X,tau=tau, method="lasso",
             lambda = cv_qr_penalty(y-i*D,X,tau=tau,penalty,CV_fold))
  }
  beta=matrix(lasso$coefficients,ncol = 1)
  e=y-i*D-cbind(1,X)%*%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_alpha[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_qr_50_2011=DML_IVQR(y,D,X,Z,tau=0.5,grid,core=4)
DML_qr_50_cv=DML_IVQR(y,D,X,Z,tau=0.5,seq(-1,3,length=41),core=4,CV=TRUE)
summary(Oracle_qr50)
```

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

```
summary(DML_qr_50_2011)
```

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

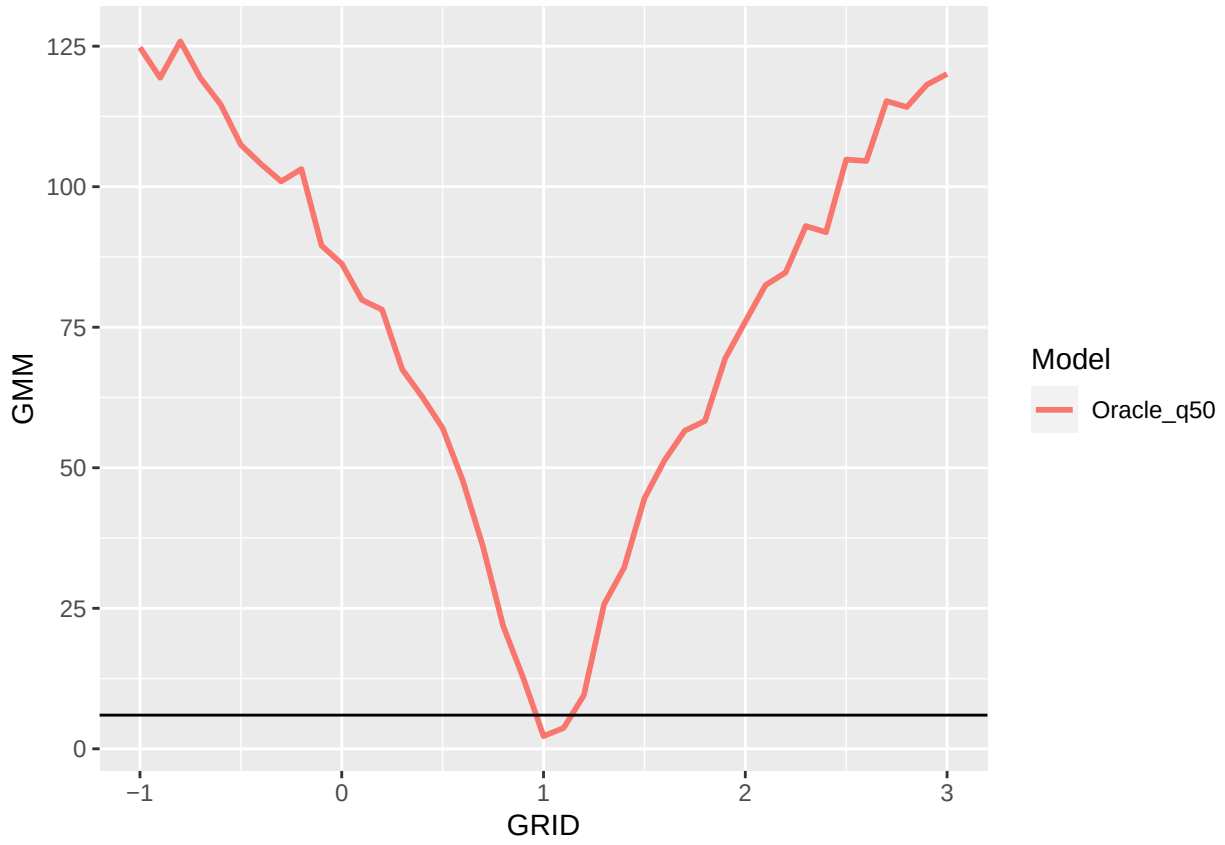
```
summary(DML_qr_50_cv)
```

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

2.3 Weak-Instrument Robust Inference with Oracle Model

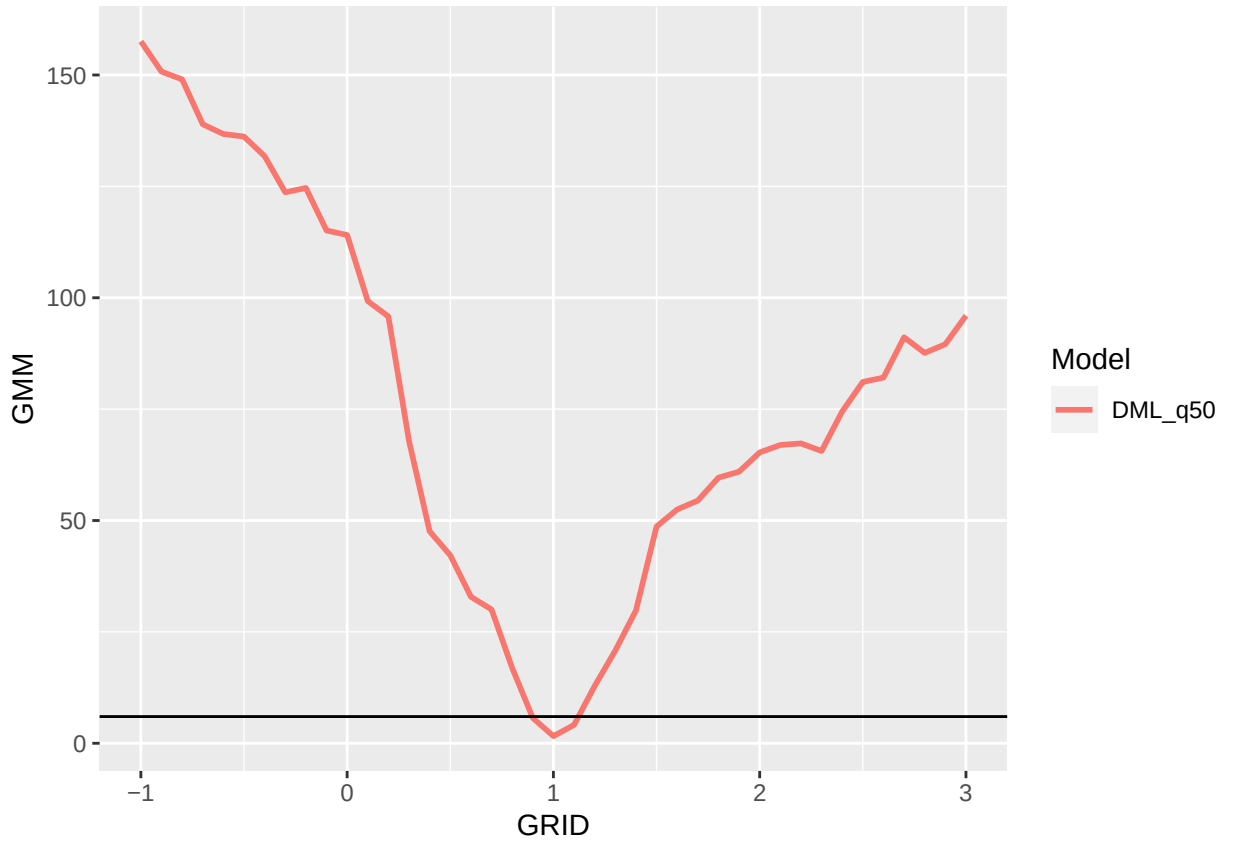
```
library(ggplot2)

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
```



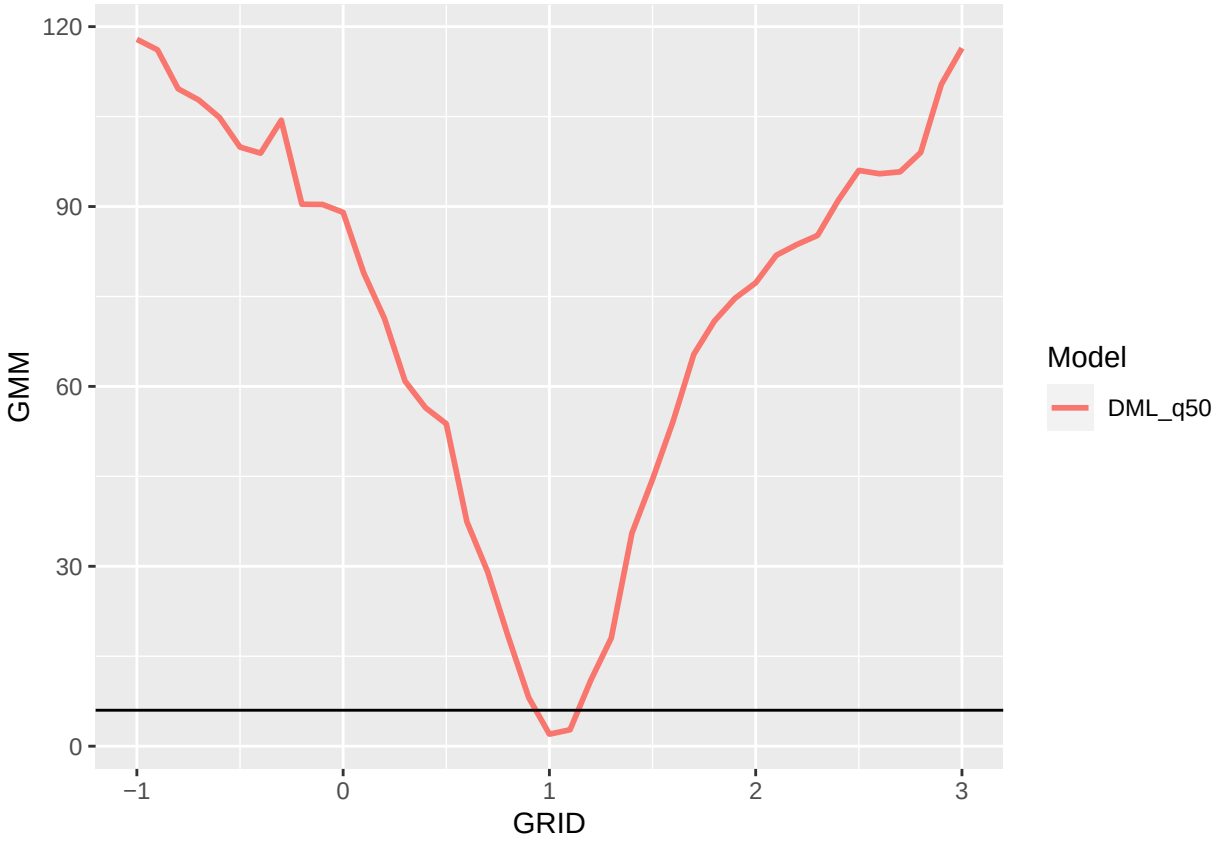
2.4 Weak-Instrument Robust Inference with DML-IVQR(Belloni and Chernozhukov (2011))

```
GMM=unlist(DML_qr_50_2011[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
```



2.4 Weak-Instrument Robust Inference with DML-IVQR (cross-validation-based λ)

```
GMM=unlist(DML_qr_50_cv[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