Package ‘cqrReg’

April 8, 2015

Type Package
Title Quantile, Composite Quantile Regression and Regularized Versions
Version 1.2
Date 2015-04-07
Author Jueyu Gao & Linglong Kong
Maintainer Jueyu Gao <jueyu@ualberta.ca>
Description Estimate quantile regression(QR) and composite quantile regression (cqr) and with adaptive lasso penalty using interior point (IP), majorize and minimize(MM), coordinate descent (CD), and alternating direction method of multipliers algorithms(ADMM).
License GPL (>= 2)
Depends Rcpp (>= 0.10.0),quantreg,R (>= 2.6)
LinkingTo Rcpp,RcppArmadillo
NeedsCompilation yes
Repository CRAN
Date/Publication 2015-04-08 07:35:09

R topics documented:
cqr.admm .................................................. 2
cqr.cd ................................................... 3
cqr.fit .................................................. 5
cqr.fit.lasso ........................................... 6
cqr.ip ................................................... 7
cqr.lasso.admm ......................................... 8
cqr.lasso.cd ........................................... 9
cqr.lasso.mm .......................................... 10
cqr.mm .................................................. 11
CQRADMMCPP .......................................... 13
CQRCDMCPP .......................................... 13
CQRMMCPP .......................................... 13
CQRPADMMCPP ...................................... 13
CQRPCDCPP .......................................... 14
Composite Quantile regression (cqr) use Alternating Direction Method of Multipliers (ADMM) algorithm.

Description

Composite quantile regression (cqr) find the estimated coefficient which minimize the absolute error for various quantile level. The problem is well suited to distributed convex optimization and is based on Alternating Direction Method of Multipliers (ADMM) algorithm.

Usage

cqr.admm(X,y,tau,rho,beta, maxit, toler)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>the design matrix</td>
</tr>
<tr>
<td>y</td>
<td>response variable</td>
</tr>
<tr>
<td>tau</td>
<td>vector of quantile level</td>
</tr>
<tr>
<td>rho</td>
<td>augmented Lagrangian parameter</td>
</tr>
<tr>
<td>beta</td>
<td>initial value of estimate coefficient (default naive guess by least square estimation)</td>
</tr>
<tr>
<td>maxit</td>
<td>maxim iteration (default 200)</td>
</tr>
<tr>
<td>toler</td>
<td>the tolerance critical for stop the algorithm (default 1e-3)</td>
</tr>
</tbody>
</table>
Value

A list structure is with components

- beta: the vector of estimated coefficient
- b: intercept

Note

cqr.admm(x, y, tau) work properly only if the least square estimation is good.

References


Examples

```r
set.seed(1)
n=100
p=2
a=rnorm(n*p, mean = 1, sd =1)
x=matrix(a,n,p)
beta=rnorm(p,1,1)
beta=matrix(beta,p,Q)
y=x%*%betaMmatrix(rnorm(n,P.Q,Q),n,Q)
tau=Q:5/6
# x is 1000*10 matrix, y is 1000*1 vector, beta is 10*1 vector
cqr.admm(x,y,tau)
```

---

**cqr.cd**

Composite Quantile Regression (cqr) use Coordinate Descent (cd) Algorithms

Description

Composite quantile regression (cqr) find the estimated coefficient which minimize the absolute error for various quantile level. The algorithm base on greedy coordinate descent and Edgeworth's for ordinary $l_1$ regression.

Usage

```r
cqr.cd(X, y, tau, beta, maxit, toler)
```
Arguments

- **x**: the design matrix
- **y**: response variable
- **tau**: vector of quantile level
- **beta**: initial value of estimate coefficient (default naive guess by least square estimation)
- **maxit**: maxim iteration (default 200)
- **tolar**: the tolerance critical for stop the algorithm (default 1e-3)

Value

- **a list** structure is with components
  - **beta**: the vector of estimated coefficient
  - **b**: intercept

Note

- `cqr.cd(x,y,tau)` work properly only if the least square estimation is good.

References


Examples

```r
set.seed(Q)
n=QPP
p=QPP
a=rnorm(n*p, mean = Q, sd =Q)
x=matrix(a,n,p)
beta=rnorm(p,1,1)
b=matrix(beta,p,1)
y=x%*%beta+matrix(rnorm(n,0,1),n,1)
tau=1:5/6
# x is 1000*10 matrix, y is 1000*1 vector, beta is 10*1 vector
cqr.cd(x,y,tau)
```
cqr.fit

Composite Quantile Regression (cqr) model fitting

Description

Composite quantile regression (cqr) find the estimated coefficient which minimize the absolute error for various quantile level. High level function for estimating parameter by composite quantile regression.

Usage

cqr.fit(x,y,tau,beta,method,maxit,toler,rho)

Arguments

x the design matrix
y response variable
tau vector of quantile level
method "mm" for majorize and minimize method,"cd" for coordinate descent method,"admm" for Alternating method of multipliers method,"ip" for interior point method
rho augmented Lagrangian parameter
beta initial value of estimate coefficient (default naive guess by least square estimation)
maxit maxim iteration (default 200)
toler the tolerance critical for stop the algorithm (default 1e-3)

Value

a list structure is with components

beta the vector of estimated coefficient
b intercept

Note

cqr.fit(x,y,tau) work properly only if the least square estimation is good. Interior point method is done by quantreg.
**Description**

Composite quantile regression (cqr) find the estimated coefficient which minimize the absolute error for various quantile level. High level function for estimating and selecting parameter by composite quantile regression with adaptive lasso penalty.

**Usage**

cqr.fit.lasso(x,y,tau,lambda,beta,method,maxit,toler,rho)

**Arguments**

- **x**: the design matrix
- **y**: response variable
- **tau**: vector of quantile level
- **method**: "mm" for majorize and minimize method,"cd" for coordinate descent method, "admm" for Alternating method of multipliers method
- **lambda**: The constant coefficient of penalty function. (default lambda=1)
- **rho**: augmented Lagrangian parameter
- **beta**: initial value of estimate coefficient (default naive guess by least square estimation)
- **maxit**: maxim iteration (default 200)
- **toler**: the tolerance critical for stop the algorithm (default 1e-3)

**Value**

* a list structure is with components
  - **beta**: the vector of estimated coefficient
  - **b**: intercept

**Note**

cqr.fit.lasso(x,y,tau) work properly only if the least square estimation is good.
Composite Quantile Regression (cqr) use Interior Point (ip) Method

Description

The function use the interior point method from quantreg to solve the quantile regression problem.

Usage

cqr.ip(X, y, tau)

Arguments

X the design matrix
y response variable
tau vector of quantile level

Value

a list structure is with components
beta the vector of estimated coefficient
b intercept

Note

Need to install quantreg package from CRAN.

References


Examples

set.seed(1)
n=100
p=2
a=rnorm(n*p, mean = 1, sd =1)
x=matrix(a,n,p)
b=matrix(rnorm(p,1,1),p,1)
y=x%*%beta+matrix(rnorm(n,0.1,1),n,1)
tau=1:5/6
# x is 1000*10 matrix, y is 1000*1 vector, beta is 10*1 vector
# you should install quantreg first to run following command
#cqr.ip(x,y,tau)
**cqr.lasso.admm**

*Composite Quantile Regression (cqr) with Adaptive Lasso Penalty (lasso) use Alternating Direction Method of Multipliers (ADMM) algorithm*

**Description**

The adaptive lasso parameter base on the estimated coefficient without penalty function. Composite quantile regression find the estimated coefficient which minimize the absolute error for various quantile level. The problem is well suited to distributed convex optimization and is based on Alternating Direction Method of Multipliers (ADMM) algorithm.

**Usage**

```r
cqr.lasso.admm(x, y, tau, lambda, rho, beta, maxit)
```

**Arguments**

- `x`: the design matrix
- `y`: response variable
- `tau`: vector of quantile level
- `lambda`: The constant coefficient of penalty function. (default lambda=1)
- `rho`: augmented Lagrangian parameter
- `beta`: initial value of estimate coefficient (default naive guess by least square estimation)
- `maxit`: maxim iteration (default 200)

**Value**

A `list` structure is with components

- `beta`: the vector of estimated coefficient
- `b`: intercept

**Note**

`cqr.lasso.admm(x, y, tau)` work properly only if the least square estimation is good.

**References**


Examples

```r
set.seed(1)
n=100
p=2
a=2*rnorm(n*2*p, mean = 1, sd =1)
x=matrix(a,n,2*p)
beta=2*rnorm(p,1,1)
beta=rbind(matrix(beta,p,1),matrix(0,p,1))
y=x%*%beta-matrix(rnorm(n,0,1),n,1)
tau=1:5/6
# x is 1000*20 matrix, y is 1000*1 vector, beta is 20*1 vector with last ten zero value elements.
cqr.lasso.admm(x,y,tau)
```

---

**cqr.lasso.cd**

*Composite Quantile Regression (cqr) with Adaptive Lasso Penalty (lasso) use Coordinate Descent (cd) Algorithms*

Description

The adaptive lasso parameter base on the estimated coefficient without penalty function. Composite quantile regression find the estimated coefficient which minimize the absolute error for various quantile level. The algorithm base on greedy coordinate descent and Edgeworth's for ordinary $l_1$ regression.

Usage

```r
cqr.lasso.cd(X,y,tau,lambda,beta,maxit,toler)
```

Arguments

- **X** the design matrix
- **y** response variable
- **tau** vector of quantile level
- **lambda** The constant coefficient of penalty function. (default lambda=1)
- **beta** initial value of estimate coefficient (default naive guess by least square estimation)
- **maxit** maxim iteration (default 200)
- **toler** the tolerance critical for stop the algorithm (default 1e-3)

Value

A list structure is with components

- **beta** the vector of estimated coefficient
- **b** intercept
Note

cqr.lasso.cd(x,y,tau) work properly only if the least square estimation is good.

References


Examples

```r
set.seed(1)
n=100
p=2
a=2*rnorm(n*2*p, mean = 1, sd =1)
x=matrix(a,n,2*p)
beta=2*rnorm(p,1,1)
beta=rbind(matrix(beta,p,1),matrix(0,p,1))
y=x%*%beta+matrix(rnorm(n,0,1),n,1)
tau=1:5/6
# x is 1000*20 matrix, y is 1000*1 vector, beta is 20*1 vector with last ten zero value elements.
cqr.lasso.cd(x,y,tau)
```

---

`cqr.lasso.mm` *Composite Quantile Regression (cqr) with Adaptive Lasso Penalty (lasso) use Majorize and Minimize (mm) Algorithm*

Description

The adaptive lasso penalty parameter base on the estimated coefficient without penalty function. Composite quantile regression find the estimated coefficient which minimize the absolute error for various quantile level. The algorithm majorizing the objective function by a quadratic function followed by minimizing that quadratic.

Usage

`cqr.lasso.mm(X,y,tau,lambda,beta,maxit, toler)`

Arguments

- `X` the design matrix
- `y` response variable
- `tau` vector of quantile level
- `lambda` The constant coefficient of penalty function. (default lambda=1)
- `beta` initial value of estimate coefficient (default naive guess by least square estimation)
maxit maxim iteration (default 200)
toler the tolerance critical for stop the algorithm (default 1e-3)

Value

a list structure is with components

beta the vector of estimated coefficient
b intercept for various quantile level

Note
cqr.lasso.mm(x,y,tau) work properly only if the least square estimation is good.

References


Examples

```
set.seed(1)
n=100
p=2
a=2*runnorm(n*2*p, mean = 1, sd =1)
x=matrix(a,n,2*p)
beta=2*runnorm(2,1,1)
beta=rbind(matrix(beta,p,1),matrix(0,p,1))
y=x%*%beta-matrix(rnorm(n,0,1,1),n,1)
tau=1:5/6
# x is 1000*20 matrix, y is 1000*1 vector, beta is 20*1 vector with last ten zero value elements.
cqr.lasso.mm(x,y,tau)
```
Arguments

- `x` the design matrix
- `y` response variable
- `tau` vector of quantile level
- `beta` initial value of estimate coefficient (default naive guess by least square estimation)
- `maxit` maxim iteration (default 200)
- `toler` the tolerance critical for stop the algorithm (default 1e-3)

Value

- A `list` structure is with components
  - `beta` the vector of estimated coefficient
  - `b` intercept for various quantile level

Note

cqr.mm(x,y,tau) work properly only if the least square estimation is good.

References


Examples

```r
set.seed(1)
n=100
p=2
a=rnorm(n*p, mean = 1, sd =1)
x=matrix(a,n,p)
beta=rnorm(p,1,1)
beta=matrix(beta,p,1)
y=x%*%beta-matrix(rnorm(n,0.1,1),n,1)
tau=1:5/6
# x is 1000*10 matrix, y is 1000*1 vector, beta is 10*1 vector
cqr.mm(x,y,tau)
```
<table>
<thead>
<tr>
<th>Description</th>
<th>Composite Quantile regression (cqr) use Alternating Direction Method of Multipliers (ADMM) algorithm core computational part</th>
</tr>
</thead>
<tbody>
<tr>
<td>CQRADMMCPP</td>
<td>Composite Quantile Regression (cqr) use Coordinate Descent (cd) Algorithms core computational part</td>
</tr>
<tr>
<td>Description</td>
<td>Composite Quantile regression (cqr) find the estimated coefficient which minimize the absolute error for various quantile level. The algorithm base on greedy coordinate descent and Edgeworth’s for ordinary $l_1$ regression.</td>
</tr>
<tr>
<td>CQRMMCPP</td>
<td>Composite Quantile Regression (cqr) use Majorize and Minimize (mm) Algorithm core computational part</td>
</tr>
<tr>
<td>Description</td>
<td>Composite quantile regression find the estimated coefficient which minimize the absolute error for various quantile level. The algorithm majorizing the objective function by a quadratic function followed by minimizing that quadratic.</td>
</tr>
<tr>
<td>CQRPADMMCPP</td>
<td>Composite Quantile Regression (cqr) with Adaptive Lasso Penalty (lasso) use Alternating Direction Method of Multipliers (ADMM) algorithm core computational part</td>
</tr>
<tr>
<td>Description</td>
<td>The adaptive lasso parameter base on the estimated coefficient without penalty function. Composite quantile regression find the estimated coefficient which minimize the absolute error for various quantile level. The problem is well suited to distributed convex optimization and is based on Alternating Direction Method of Multipliers (ADMM) algorithm.</td>
</tr>
</tbody>
</table>
**Description**

The adaptive lasso penalty parameter base on the estimated coefficient without penalty function. Composite quantile regression find the estimated coefficient which minimize the absolute error for various quantile level. The algorithm base on greedy coordinate descent and Edgeworth’s for ordinary $l_1$ regression.

**Usage**

```r
QR.admm(X,y,tau,rho,beta, maxit, toler)
```
Arguments

\begin{align*}
x & \quad \text{the design matrix} \\
y & \quad \text{response variable} \\
\tau & \quad \text{quantile level} \\
\rho & \quad \text{augmented Lagrangian parameter} \\
\beta & \quad \text{initial value of estimate coefficient (default naive guess by least square estimation)} \\
\maxit & \quad \text{maxim iteration (default 200)} \\
\toler & \quad \text{the tolerance critical for stop the algorithm (default 1e-3)}
\end{align*}

Value

\begin{itemize}
  \item \textbf{a list} structure is with components
  \begin{itemize}
    \item \textbf{beta} \quad \text{the vector of estimated coefficient}
    \item \textbf{b} \quad \text{intercept}
  \end{itemize}
\end{itemize}

Note

QR.admm(x,y,tau) work properly only if the least square estimation is good.

References


Examples

\begin{verbatim}
set.seed(1)
n=100
p=2
a=rnorm(n*p, mean = 1, sd =1)
x=matrix(a,n,p)
beta=rnorm(p,1,1)
beta=matrix(beta,p,1)
y=x%*%beta+matrix(rnorm(n,0.1,1),n,1)
# x is 1000x10 matrix, y is 1000x1 vector, beta is 10x1 vector
QR.admm(x,y,0.1)
\end{verbatim}
Quantile Regression (QR) use Coordinate Descent (cd) Algorithms

Description

The algorithm base on greedy coordinate descent and Edgeworth’s for ordinary $l_1$ regression.

Usage

```r
QR.cd(x, y, tau, beta, maxit, toler)
```

Arguments

- `x`: the design matrix
- `y`: response variable
- `tau`: quantile level
- `beta`: initial value of estimate coefficient (default naive guess by least square estimation)
- `maxit`: maxim iteration (default 200)
- `toler`: the tolerance critical for stop the algorithm (default 1e-3)

Value

A list structure is with components

- `beta`: the vector of estimated coefficient
- `b`: intercept

Note

`QR.cd(x, y, tau)` work properly only if the least square estimation is good.

References


Examples

```r
set.seed(1)
n=100
p=2
a=rnorm(n*p, mean = 1, sd =1)
x=matrix(a,n,p)
beta=rnorm(p,1,1)
beta=matrix(beta,p,1)
```
**QR.ip**

\[ y = x \times \beta \times \text{matrix}(rnorm(n,0,1),n,1) \]

# x is 1000x10 matrix, y is 1000x1 vector, beta is 10x1 vector
QR.cd(x,y,0.1)

---

**Description**

The function uses the interior point method from quantreg to solve the quantile regression problem.

**Usage**

`QR.ip(X,y,tau)`

**Arguments**

- **X**: the design matrix
- **y**: the response variable
- **tau**: quantile level

**Value**

A list structure is with components

- **beta**: the vector of estimated coefficient
- **b**: intercept

**Note**

Need to install quantreg package from CRAN.

**References**


**Examples**

```r
set.seed(Q)
n=QPP
p=R
a=rnorm(n*p, mean = Q, sd = Q)
x=matrix(a,n,p)
beta=rnorm(p,Q,Q)
beta=matrix(beta,p,Q)
y=x%*%beta-matrix(rnorm(n,P.Q,Q),n,Q)
```
QR.lasso.admm

Quantile Regression (QR) with Adaptive Lasso Penalty (lasso) use Alternating Direction Method of Multipliers (ADMM) algorithm

Description

The adaptive lasso parameter base on the estimated coefficient without penalty function. The problem is well suited to distributed convex optimization and is based on Alternating Direction Method of Multipliers (ADMM) algorithm.

Usage

QR.lasso.admm(X,y,tau,lambda,rho,beta,maxit)

Arguments

x the design matrix
y response variable
tau quantile level
lambda The constant coefficient of penalty function. (default lambda=1)
rho augmented Lagrangian parameter
beta initial value of estimate coefficient (default naive guess by least square estimation)
maxit maxim iteration (default 200)

Value

a list structure is with components
beta the vector of estimated coefficient
b intercept

Note

QR.lasso.admm(x,y,tau) work properly only if the least square estimation is good.

References


Examples

```r
set.seed(1)
n=100
p=2
a=2*runif(n,2*p, mean = 1, sd =1)
x=matrix(a,n,2*p)
beta=2*runif(p,1,1)
beta=rrbind(matrix(beta,p,1),matrix(0,p,1))
y=x%*%beta
# x is 1000*20 matrix, y is 1000*1 vector, beta is 20*1 vector with last ten zero value elements.
QR.lasso.admm(x,y,0.1)
```

QR.lasso.cd  Quantile Regression (QR) with Adaptive Lasso Penalty (lasso) use Coordinate Descent (cd) Algorithms

Description

The adaptive lasso parameter base on the estimated coefficient without penalty function. The algorithm base on greedy coordinate descent and Edgeworth’s for ordinary $l_1$ regression. As explored by Tong Tong Wu and Kenneth Lange.

Usage

```r
QR.lasso.cd(x,y,tau,lambda,beta,maxit,toler)
```

Arguments

- `x`: the design matrix
- `y`: response variable
- `tau`: quantile level
- `lambda`: The constant coefficient of penalty function. (default lambda=1)
- `beta`: initial value of estimate coefficient (default naive guess by least square estimation)
- `maxit`: maxim iteration (default 200)
- `toler`: the tolerance critical for stop the algorithm (default 1e-3)

Value

A list structure is with components

- `beta`: the vector of estimated coefficient
- `b`: intercept

Note

QR.lasso.cd(x,y,tau) work properly only if the least square estimation is good.
References


Examples

```r
set.seed(Q)
n=QPP
p=R
a=R*rnorm(n*R*p, mean = Q, sd =Q)
x=matrix(a,n,R*p)
beta=R*rnorm(p,Q,Q)
beta=rbind(matrix(beta,p,Q),matrix(P,p,Q))
y=x%*%betaMmatrix(rnorm(n,P.Q,Q),n,Q)
```

 qr.lasso.cd(x,y,P.Q)  

**QR.lasso.ip**  

*Quantile Regression (QR) with Adaptive Lasso Penalty (lasso) use Interior Point (ip) Method*

**Description**

The function use the interior point method from quantreg to solve the quantile regression problem.

**Usage**

```r
QR.lasso.ip(X,y,tau,lambda)
```

**Arguments**

- `X`  
  the design matrix
- `y`  
  response variable
- `tau`  
  quantile level
- `lambda`  
  The constant coefficient of penalty function. (default lambda=1)

**Value**

`a list` structure is with components

- `beta`  
  the vector of estimated coefficient
- `b`  
  intercept
- `lambda`  
  The constant coefficient of penalty function. (default lambda=1)
Note

Need to install quantreg package from CRAN.

References


Examples

```r
set.seed(1)
n=100
p=2
a=2*rnorm(n*2*p, mean = 1, sd =1)
x=matrix(a,n,2*p)
beta=2*rnorm(p,1,1)
beta=cbind(matrix(beta,p,1),matrix(0,p,1))
y=x%*%beta + matrix(rnorm(n,0,1),n,1)
# x is 1000*20 matrix, y is 1000*1 vector, beta is 20*1 vector with last ten zero value elements.
# you should install Rmosek first to run following command
#QR.lasso.ip(x,y,0.1)
```

QR.lasso.mm

Quantile Regression (QR) with Adaptive Lasso Penalty (lasso) use Majorize and Minimize (mm) algorithm

Description

The adaptive lasso parameter base on the estimated coefficient without penalty function. The algorithm majorizing the objective function by a quadratic function followed by minimizing that quadratic.

Usage

```r
QR.lasso.mm(X,y,tau,lambda,beta,maxit,toler)
```

Arguments

- `x` the design matrix.
- `y` response variable.
- `tau` quantile level.
- `lambda` The constant coefficient of penalty function. (default lambda=1)
- `beta` initial value of estimate coefficient.(default naive guess by least square estimation)
- `maxit` maxim iteration. (default 200)
- `toler` the tolerance critical for stop the algorithm. (default 1e-3)
Value

A list structure is with components

beta the vector of estimated coefficient
b intercept

Note

QR.lasso.mm(x,y,tau) work properly only if the least square estimation is good.

References


Examples

```r
set.seed(Q)
n=QPP
p=R
a=R*rnorm(n*R*p, mean = Q, sd =Q)
x=matrix(a,n,R*p)
beta=R*rnorm(p,Q,Q)
beta=rbind(matrix(beta,p,Q),matrix(P,p,Q))
y=x%*%betaMmatrix(rnorm(n,P.Q,Q),n,Q)
# x is QPPP*RP matrix, y is QPPP*Q vector, beta is RP*Q vector with last ten zero value elements.
QR.lasso.mm(x,y,0.1)
```

QR.mm

Quantile Regression (QR) use Majorize and Minimize (mm) algorithm

Description

The algorithm majorizing the objective function by a quadratic function followed by minimizing that quadratic.

Usage

`QR.mm(X,y,tau,beta,maxit,toler)`

Arguments

- `X` the design matrix
- `y` response variable
- `tau` quantile level
- `beta` initial value of estimate coefficient (default naive guess by least square estimation)
- `maxit` maxim iteration (default 200)
- `toler` the tolerance critical for stop the algorithm (default 1e-3)
Value

A list structure is with components

- `beta` the vector of estimated coefficient
- `b` intercept

Note

QR.mm(x,y,tau) work properly only if the least square estimation is good.

References

David R. Hunter and Kenneth Lange. Quantile Regression via an MM Algorithm, *Journal of Computational and Graphical Statistics*, 9, Number 1, Page 60–77

Examples

```r
set.seed(Q)
n=QPP
p=R
a=rnorm(n*p, mean = Q, sd =Q)
x=matrix(a,n,p)
beta=rnorm(p,Q,Q)
beta=matrix(beta,p,Q)
y=x%*%betaMmatrix(rnorm(n,P.Q,Q),n,Q)
```

QRADMMCPP

Quantile Regression (QR) use Alternating Direction Method of Multipliers (ADMM) algorithm core computational part

Description

The problem is well suited to distributed convex optimization and is based on Alternating Direction Method of Multipliers (ADMM) algorithm.

QRCDCPP

Quantile Regression (QR) use Coordinate Descent (cd) Algorithms core computational part

Description

The algorithm base on greedy coordinate descent and Edgeworth’s for ordinary $\ell_1$ regression.
**qrfit**

*Quantile Regression (qr) model fitting*

**Description**

High level function for estimating parameters by quantile regression

**Usage**

```r
garfit(X, y, tau, beta, method, maxit, toler, rho)
```

**Arguments**

- `x`: the design matrix
- `y`: response variable
- `tau`: quantile level
- `rho`: augmented Lagrangian parameter
- `beta`: initial value of estimate coefficient (default naive guess by least square estimation)
- `maxit`: maxim iteration (default 200)
- `toler`: the tolerance critical for stop the algorithm (default 1e-3)

**Value**

A list structure is with components

- `beta`: the vector of estimated coefficient
- `b`: intercept

**Note**

`qrfit(x,y,tau)` work properly only if the least square estimation is good. Interior point method is done by quantreg.
**qrfit.lasso**

**Quantile Regression (qr) with Adaptive Lasso Penalty (lasso)**

**Description**

High level function for estimating and selecting parameter by quantile regression with adaptive lasso penalty.

**Usage**

```r
qrfit.lasso(x, y, tau, lambda, beta, method, maxit, toler, rho)
```

**Arguments**

- `x`: the design matrix
- `y`: response variable
- `tau`: quantile level
- `lambda`: The constant coefficient of penalty function. (default lambda=1)
- `rho`: augmented Lagrangian parameter
- `beta`: initial value of estimate coefficient (default naive guess by least square estimation)
- `maxit`: maxim iteration (default 200)
- `toler`: the tolerance critical for stop the algorithm (default 1e-3)

**Value**

a list structure is with components

- `beta`: the vector of estimated coefficient
- `b`: intercept

**Note**

`qrfit.lasso(x,y,tau)` work properly only if the least square estimation is good. Interior point method is done by quantreg.
<table>
<thead>
<tr>
<th>QRPMMCPP</th>
<th>Quantile Regression (QR) use Majorize and Minimize (mm) algorithm core computational part</th>
</tr>
</thead>
</table>

**Description**

The algorithm majorizing the objective function by a quadratic function followed by minimizing that quadratic.

<table>
<thead>
<tr>
<th>QRPADEMMCPP</th>
<th>Quantile Regression (QR) with Adaptive Lasso Penalty (lasso) use Alternating Direction Method of Multipliers (ADMM) algorithm core computational part</th>
</tr>
</thead>
</table>

**Description**

The adaptive lasso parameter base on the estimated coefficient without penalty function. The problem is well suited to distributed convex optimization and is based on Alternating Direction Method of Multipliers (ADMM) algorithm.

<table>
<thead>
<tr>
<th>QRPCDCPP</th>
<th>Quantile Regression (QR) with Adaptive Lasso Penalty (lasso) use Coordinate Descent (cd) Algorithms core computational part</th>
</tr>
</thead>
</table>

**Description**

The adaptive lasso parameter base on the estimated coefficient without penalty function. The algorithm base on greedy coordinate descent and Edgeworth’s for ordinary $l_1$ regression. As explored by Tong Tong Wu and Kenneth Lange.

<table>
<thead>
<tr>
<th>QRPMMCPP</th>
<th>Quantile Regression (QR) with Adaptive Lasso Penalty (lasso) use Majorize and Minimize (mm) algorithm core computational part</th>
</tr>
</thead>
</table>

**Description**

The adaptive lasso parameter base on the estimated coefficient without penalty function. The algorithm majorizing the objective function by a quadratic function followed by minimizing that quadratic.
Index

*Topic ADMM
  cqr.admm, 2
  cqr.lasso.admm, 8
  CQRADMMCPP, 13
  CQRPADMMCPP, 13
  QR.admm, 14
  QR.lasso.admm, 18
  QRADMMCPP, 23
  QRPADMMCPP, 26

*Topic CD
  cqr.cd, 3
  cqr.lasso.cd, 9
  CQRCDCPP, 13
  CQRPCDCPP, 14
  QR.cd, 16
  QR.lasso.cd, 19
  QRCDCPP, 23
  QRPDCcpp, 26

*Topic Composite quantile regression
  cqr.fit, 5
  cqr.fit.lasso, 6

*Topic IP
  cqr.ip, 7
  QR.ip, 17
  QR.lasso.ip, 20

*Topic MM
  cqr.lasso.mm, 10
  cqr.mm, 11
  CQRMMCPP, 13
  CQRPMMCPP, 14
  QR.lasso.mm, 21
  QR.mm, 22
  QRMMCPP, 26
  QRPMMCPP, 26

*Topic Quantile regression
  qrfit, 24
  qrfit.lasso, 25

  cqr.admm, 2
  cqr.cd, 3
  cqr.fit, 5
  cqr.fit.lasso, 6
  cqr.ip, 7
  cqr.lasso.admm, 8
  cqr.lasso.cd, 9
  cqr.lasso.mm, 10
  cqr.mm, 11
  CQRADMMCPP, 13
  CQRCDCPP, 13
  CQRMMCPP, 13
  CQRPCDCPP, 14
  CQRPMDCPP, 14

  list, 3–9, 11, 12, 15–20, 22–25
  QR.admm, 14
  QR.cd, 16
  QR.ip, 17
  QR.lasso.admm, 18
  QR.lasso.cd, 19
  QR.lasso.ip, 20
  QR.lasso.mm, 21
  QR.mm, 22
  QRADMMCPP, 23
  QRCDCPP, 23
  qrfit, 24
  qrfit.lasso, 25
  QRMCPP, 26
  QRPADMMCPP, 26
  QRPDCcpp, 26
  QRPMMCPP, 26

27