Package ‘cluscov’

October 12, 2022

Type Package

Title Clustered Covariate Regression

Version 1.1.0

Date 2019-05-31

Author Emmanuel S Tsyawo [aut, cre], Abdul-Nasah Soale [aut]

Maintainer Emmanuel S Tsyawo <estsyawo@temple.edu>

Description Clustered covariate regression enables estimation and inference in both linear and non-linear models with linear predictor functions even when the design matrix is column rank deficient. Routines in this package implement algorithms in Soale and Tsyawo (2019) <doi:10.13140/RG.2.2.32355.81441>.

License GPL-2

Encoding UTF-8

LazyData true

Imports quantreg, MASS, stats, utils

RoxygenNote 6.1.1

NeedsCompilation yes

Repository CRAN

Date/Publication 2019-06-04 12:30:07 UTC

R topics documented:

`CCRls` *Sequential CCR*

Description

`CCRls` runs regressions with potentially more covariates than observations. See `c_chmod()` for the list of models supported.

Usage

```
CCRls(Y, X, kap = 0.1, modclass = "lm", tol = 1e-06, reltol = TRUE,
  rndcov = NULL, report = NULL, ...)
```

Arguments

`Y`	vector of dependent variable Y
`X`	design matrix (without intercept)
`kap`	maximum number of parameters to estimate in each active sequential step, as a fraction of the less of total number of observations n or number of covariates p. i.e. $min(n, p)$
`modclass`	a string denoting the desired the class of model. See c_chmod for details.
`tol`	level of tolerance for convergence; default `tol=1e-6`
`reltol`	a logical for relative tolerance instead of level. Defaults at TRUE
`rndcov`	seed for randomising assignment of covariates to partitions; default `NULL`
`report`	number of iterations after which to report progress; default `NULL`
`...`	additional arguments to be passed to the model

Value

`betas` parameter estimates (intercept first),

`iter` number of iterations,

`dev` increment in the objective function value at convergence

`fval` objective function value at convergence

Examples

```
set.seed(14) #Generate data
N = 1000; (bets = rep(-2:2,4)); p = length(bets); X = matrix(rnorm(N*p),N,p)
Y = cbind(1,X)%*%matrix(c(0.5,bets),ncol = 1)
CCRls(Y,X,kap=0.1,modclass="lm",tol=1e-6,reltol=TRUE,rndcov=NULL,report=8)
```

CCRls.coord — *Linear regression via coordinate descent with covariate clustering*

Description

This function is a wrapper for `linrclus`. It requires less input.

Usage

```
CCRls.coord(Y, X, k, nC = 1, ...)
```

Arguments

`Y`	vector of outcome variable
`X`	matrix of covariates. Should not include 1's for the intercept
`k`	number of clusters
`nC`	first nC-1 covariates in X not to cluster. Must be at least 1 for the intercept
`...`	additional parameters to be passed to lm

Value

`mobj` the low dimension lm regression object

`clus` cluster assignments of covariates (excluding the first nC covariates - including the intercept 1)

Examples

```
set.seed(14) #Generate data
N = 1000; (bets = rep(-2:2,4)); p = length(bets); X = matrix(rnorm(N*p),N,p)
Y = cbind(1,X)%*%matrix(c(0.5,bets),ncol = 1)
CCRls.coord(Y,X,k=5,nC=1)
```

CCRseqk *Sequential CCR with k clusters*

Description

CCRseqk runs regressions with potentially more covariates than observations with k clusters. See c_chmod() for the list of models supported.

Usage

```
CCRseqk(Y, X, k, nC = 1, kap = 0.1, modclass = "lm", tol = 1e-06,
  reltol = TRUE, rndcov = NULL, report = NULL, ...)
```

Arguments

Y	vector of dependent variable Y
X	design matrix (without intercept)
k	number of clusters
nC	first nC-1 columns in X not to cluster
kap	maximum number of parameters to estimate in each active sequential step, as a fraction of the less of total number of observations n or number of covariates p. i.e. $min(n, p)$
modclass	a string denoting the desired the class of model. See c_chmod for details.
tol	level of tolerance for convergence; default tol=1e-6
reltol	a logical for relative tolerance instead of level. Defaults at TRUE
rndcov	seed for randomising assignment of covariates to partitions; default NULL
report	number of iterations after which to report progress; default NULL
...	additional arguments to be passed to the model

Value

a list of objects

- mobj low dimensional model object of class lm, glm, or rq (depending on modclass)
- clus cluster assignments of covariates
- iter number of iterations
- dev decrease in the function value at convergence

Examples

```
set.seed(14) #Generate data
N = 1000; (bets = rep(-2:2,4)/2); p = length(bets); X = matrix(rnorm(N*p),N,p)
Y = cbind(1,X)%*%matrix(c(0.5,bets),ncol = 1); nC=1
zg=CCRseqk(Y,X,k=5,nC=nC,kap=0.1,modclass="lm",tol=1e-6,reltol=TRUE,rndcov=NULL,report=8)
(del=zg$mobj$coefficients) # delta
(bets = c(del[1:nC],(del[-c(1:nC)])[zg$clus])) #construct beta
```

chmod — *Model criterion function*

Description

A generic S3 function as wrapper for internal R routines for classes of models implemented in this package. See details c_chmod for the list of classes supported.

Usage

```
chmod(object, ...)
```

Arguments

`object`	the object to be passed to the concrete class constructor `chmod`
`...`	additional paramters to be passed to the internal routine

chmod.gammainverse — *Regression - gammainverse class*

Description

A gamma regression implementation for the "gammainverse" class. It uses `glm` with the Gamma link function set to "inverse"

Usage

```
## S3 method for class 'gammainverse'
chmod(object, ...)
```

Arguments

`object`	a list of Y - outcome variable and X - design matrix of class "probit"
`...`	additional parameters to be passed to `glm`

Value

fitted model object

Examples

```
chmod(c_chmod(Y=women$height,X=women$weight,modclass="gammainverse"))
```

chmod.gammalog *Regression - gammalog class*

Description

A gamma regression implementation for the "gammalog" class. It uses glm with the Gamma link function set to "log"

Usage

```
## S3 method for class 'gammalog'
chmod(object, ...)
```

Arguments

object	a list of Y - outcome variable and X - design matrix of class "probit"
...	additional parameters to be passed to glm

Value

fitted model object

Examples

```
chmod(c_chmod(Y=women$height,X=women$weight,modclass="gammalog"))
```

chmod.lm *Regression - lm class*

Description

A linear regression implementation for the "lm" class. It uses lm

Usage

```
## S3 method for class 'lm'
chmod(object, ...)
```

Arguments

object	a list of Y - outcome variable and X - design matrix of class "lm"
...	additional parameters to be passed to lm

Value

fitted model object

Examples

```
chmod(c_chmod(Y=women$height,X=women$weight,modclass="lm"))
```

---

chmod.logit *Regression - logit class*

Description

A logit regression implementation for the "logit" class. It uses glm with the binomial link function set to "logit"

Usage

```
## S3 method for class 'logit'
chmod(object, ...)
```

Arguments

object	a list of Y - outcome variable and X - design matrix of class "logit"
...	additional parameters to be passed to glm

Value

fitted model object

Examples

```
chmod(c_chmod(Y=women$height<=50,X=women$weight,modclass="logit"))
```

---

chmod.negbin *Regression - negbin class*

Description

A negative binomial regression implementation for the "negbin" class. It uses glm.nb

Usage

```
## S3 method for class 'negbin'
chmod(object, ...)
```

Arguments

object	a list of Y - outcome variable and X - design matrix of class "negbin"
...	additional parameters to be passed to glm.nb

Value

fitted model object

---

chmod.poissonidentity *Regression - poissonidentity class*

Description

A poisson regression implementation for the "poissonidentity" class. It uses glm with the poisson link function set to "identity"

Usage

```
## S3 method for class 'poissonidentity'
chmod(object, ...)
```

Arguments

object	a list of Y - outcome variable and X - design matrix of class "poissonidentity"
...	additional parameters to be passed to glm

Value

fitted model object

Examples

```
chmod(c_chmod(Y=women$height,X=women$weight,modclass="poissonidentity"))
```

---

chmod.poissonlog *Regression - poissonlog class*

Description

A poisson regression implementation for the "poissonlog" class. It uses glm with the poisson link function set to "log"

Usage

```
## S3 method for class 'poissonlog'
chmod(object, ...)
```

Arguments

object	a list of Y - outcome variable and X - design matrix of class "poissonlog"
...	additional parameters to be passed to glm

Value

fitted model object

Examples

```
chmod(c_chmod(Y=women$height,X=women$weight,modclass="poissonlog"))
```

---

chmod.poissonsqrt *Regression - poissonsqrt class*

Description

A poisson regression implementation for the "poissonsqrt" class. It uses `glm` with the poisson link function set to "sqrt"

Usage

```
## S3 method for class 'poissonsqrt'
chmod(object, ...)
```

Arguments

`object`	a list of Y - outcome variable and X - design matrix of class "poissonsqrt"
`...`	additional parameters to be passed to `glm`

Value

fitted model object

Examples

```
chmod(c_chmod(Y=women$height,X=women$weight,modclass="poissonsqrt"))
```

---

chmod.probit *Regression - probit class*

Description

A probit regression implementation for the "probit" class. It uses `glm` with the binomial link set to "probit"

Usage

```
## S3 method for class 'probit'
chmod(object, ...)
```

Arguments

`object`	a list of Y - outcome variable and X - design matrix of class "probit"
`...`	additional parameters to be passed to `glm`

Value

fitted model object

Examples

```
chmod(c_chmod(Y=women$height<=50,X=women$weight,modclass="probit"))
```

---

chmod.qreg — *Regression - qreg class*

Description

A quantile regression implementation for the "qreg" class. It uses `rq`

Usage

```
## S3 method for class 'qreg'
chmod(object, ...)
```

Arguments

`object`	a list of Y - outcome variable and X - design matrix of class "qreg"
`...`	additional parameters to be passed to `rq`, for example `tau`

Value

fitted model object

Examples

```
chmod(c_chmod(Y=women$height,X=women$weight,modclass="qreg"),tau=0.45)
```

c_chmod — *Concrete class constructor*

Description

A function for constructing functions for concrete classes of models for the `chmod()` family of of functions.

Usage

```
c_chmod(Y, X, modclass = "lm")
```

Arguments

`Y`	vector of the outcome variable
`X`	matrix of covariates; excepting intercepts 1's
`modclass`	the class of model. Currently, "lm" for linear regression, "logit" (logit model), "qreg" (quantile regression), "probit" (probit model), "gammainverse" (gamma with inverse link), "gammalog" (gamma with log link), "poissonlog" (poisson model with log link), "poissonidentity" (poisson with identity link), "poisson-sqrt" (poisson with sqrt link), "negbin" (negative binomial) are supported.

Value

object an object list with class attribute modclass.

dcluspar — *Clustering of vector elements*

Description

A deterministic clustering device of vector elements into k clusters

Usage

```
dcluspar(k, vec)
```

Arguments

`k`	number of clusters
`vec`	the vector of real valued elements

Value

clus integer assignment of corresponding elements in vec in up to k clusters

Examples

```
set.seed(2); (v=c(rnorm(4,0,0.5),rnorm(3,3,0.5))[sample(1:7)])
dcluspar(k=2,vec = v)
```

`goldensearch` *Golden Section Search Algorithm*

Description

Minimising a continuous univariate function using the golden section search algorithm.

Usage

```
goldensearch(fn, interval, tol = 1)
```

Arguments

`fn`	the function; should be scalar valued
`interval`	a vector containing the lower and upper bounds of search
`tol`	tolerance level for convergence

Value

a list of objects

- k: minimiser
- value: mimimum value
- iter: number of iterations before convergence
- iterfn: number of function evaluations

Examples

```
fn = function(x) (x-1)^2; goldensearch(fn=fn,interval=c(-2,3),tol=1)
```

goldopt *Integer Golden Search Minimisation*

Description

This function conducts an integer golden search minimisation of a univariate function.

Usage

```
goldopt(fn, interval, tol = 1)
```

Arguments

`fn`	function to be minimised. **fn** should return a list, with **fval** as the function value.
`interval`	a vector of length two containing the minimum and maximum interger values within which to search for the minimiser.
`tol`	the tolerance level. Defaults at 1

Value

k minimiser of `fn()`

crit the minimum

iter total number of iterations

iterfn total number of function evaluations of `fn()`

fobj an object of the function minimisation

key a logical for warning if `fobj` may not correspond to `k`

Examples

```
set.seed(14) #Generate data
N = 1000; (bets = rep(-2:2,4)); p = length(bets); X = matrix(rnorm(N*p),N,p)
Y = cbind(1,X)%*%matrix(c(0.5,bets),ncol = 1)
fn=function(k){du=CCRls.coord(Y,X,k=k,nC=1)
return(list(fval=BIC(du$mobj),obj=du))}
goldopt(fn=fn,interval=c(2,7),tol=1)
```

`linrclus` *Linear regression via coordinate descent with covariate clustering*

Description

Covariate assignment to k clusters using the coordinate descent algorithm. This function is a wrapper for the `C` function `linreg_coord_clus`

Usage

```
linrclus(Y, X, k, coefs, clus, clusmns, nC = 1, x = FALSE)
```

Arguments

`Y`	vector of outcome variable
`X`	matrix of covariates. Should not include 1's for the intercept
`k`	number of clusters
`coefs`	vector of coefficients as starting values. Should not include the intercept.
`clus`	vector of covariate cluster assignments as starting values
`clusmns`	vector k cluster parameter centers
`nC`	first nC-1 covariates in X not to cluster. Must be at least 1 for the intercept
`x`	a logical for returning the design matrix

Value

clus cluster assignments

coefs vector of coefficients as starting values

clusmns vector of cluster means

Examples

```
set.seed(14) #Generate data
N = 1000; (bets = rep(-2:2,4)); p = length(bets); X = matrix(rnorm(N*p),N,p)
Y = cbind(1,X)%*%matrix(c(0.5,bets),ncol = 1)
begin_v<- rep(NA,p)
for (j in 1:p) {
 begin_v[j] = stats::coef(lm(Y~X[,j]))[2]
}
set.seed(12); klus_obj<- kmeans(begin_v,centers = 5)
linrclus(Y,X,k=5,coefs=c(0,begin_v),clus=klus_obj$cluster,clusmns=klus_obj$centers)
```

netdat — *Construct a network design matrix*

Description

This function creates the design matrix for a latent network structure using a balanced panel

Usage

```
netdat(datf, Y, X, Wi, W = NULL, panvar, tvar, factors, scaling = TRUE,
  unicons = TRUE)
```

Arguments

`datf`	the entire data frame of balanced panel with NT rows of unit-time observations
`Y`	dependent variable in the data frame datf
`X`	the covariate(s) generating spillovers
`Wi`	other unit-varying (can be time-invariant) control variables
`W`	global variables. these are only time varying but are common to all units. eg. GDP for individual/state-level data. Note that W has to be a vector of length T so cannot be in the data frame datf
`panvar`	the panel variable eg. unique person/firm identifiers
`tvar`	time variable, eg. years
`factors`	a vector of characters of factors in the data
`scaling`	a logical indicating whether non-discrete covariates should be scaled by their standard deviations
`unicons`	a logical indicating whether to include unit-specific constant term

Value

Y vector of dependent variables

X a block matrix of spillover matrix (TN x N^2)

Wm a matrix corresponding to covariate Wi

Wf a matrix of dummies corresponding to factors

Index