Package: MADMMplasso (via r-universe)

October 11, 2026

Title Multi Variate Multi Response ADMM with Interaction Effects

Version 1.0.1

Description This system allows one to model a multi-variate, multi-response problem with interaction effects. It combines the usual squared error loss for the multi-response problem with some penalty terms to encourage responses that correlate to form groups and also allow for modeling main and interaction effects that exit within the covariates. The optimization method employed is the Alternating Direction Method of Multipliers (ADMM). The implementation is based on the methodology presented on Quachie Asenso, T., & Zucknick, M. (2023) <doi:10.48550/arXiv.2303.11155>.

Imports Matrix, MASS, Rcpp, RcppArmadillo, foreach, doParallel, class, graphics, parallel, stats, spatstat.sparse, methods

LinkingTo Rcpp, RcppArmadillo

Suggests testthat (>= 3.0.0), lintr

License GPL-3

Encoding UTF-8

RoxygenNote 7.3.2

Config/testthat/edition 3

Date 2025-10-27

Language en-GB

NeedsCompilation yes

Author Theophilus Quachie Asenso [aut], Manuela Zucknick [aut], Waldir Leoncio [aut, cre] (ORCID: <https://orcid.org/0000-0002-6719-6162>), Chi Zhang [aut]

Maintainer Waldir Leoncio <w.l.netto@medisin.uio.no>

Repository https://cran.r-universe.dev

Date/Publication 2025-10-27 14:37:35 UTC

RemoteUrl https://github.com/cran/MADMMplasso

RemoteRef HEAD

RemoteSha df57aa774c5f3c6c96477abb24c39981944b161e

Contents

admm_MADMMplasso 2
admm_MADMMplasso_cpp 4
compute_pliable 6
cv_MADMMplasso 6
generate_my_w 10
MADMMplasso 11
predict.MADMMplasso 15
sim2 16
tree_parms 16

Index **18**

`admm_MADMMplasso` *Fit the ADMM part of model for the given lambda values*

Description

This function fits a multi-response pliable lasso model over a path of regularization values.

Usage

```
admm_MADMMplasso(
  beta0,
  theta0,
  beta,
  beta_hat,
  theta,
  rho1,
  X,
  Z,
  max_it,
  W_hat,
  XtY,
  y,
  N,
  e.abs,
  e.rel,
  alpha,
  lambda,
  alph,
  svd.w,
  tree,
  my_print,
  invmat,
  gg = 0.2
)
```

Arguments

beta0	a vector of length ncol(y) of estimated beta_0 coefficients
theta0	matrix of the initial theta_0 coefficients ncol(Z) by ncol(y)
beta	a matrix of the initial beta coefficients ncol(X) by ncol(y)
beta_hat	a matrix of the initial beta and theta coefficients (ncol(X)+ncol(X) by ncol(Z)) by ncol(y)
theta	an array of initial theta coefficients ncol(X) by ncol(Z) by ncol(y)
rho1	the Lagrange variable for the ADMM which is usually included as rho in the MADMMplasso call.
X	N by p matrix of predictors
Z	N by K matrix of modifying variables. The elements of Z may represent quantitative or categorical variables, or a mixture of the two. Categorical variables should be coded by 0-1 dummy variables: for a k-level variable, one can use either k or k-1 dummy variables.
max_it	maximum number of iterations in loop for one lambda during the ADMM optimization
W_hat	N by (p+(p by nz)) of the main and interaction predictors. This generated internally when MADMMplasso is called or by using the function generate_my_w.
XtY	a matrix formed by multiplying the transpose of X by y.
y	N by D matrix of responses. The X and Z variables are centered in the function. We recommend that X and Z also be standardized before the call
N	nrow(X)
e.abs	absolute error for the ADMM
e.rel	relative error for the ADMM
alpha	mixing parameter. When the goal is to include more interactions, alpha should be very small and vice versa.
lambda	user specified lambda_3 values.
alph	an overrelaxation parameter in [1, 1.8]. The implementation is borrowed from Stephen Boyd's MATLAB code
svd.w	singular value decomposition of W
tree	The results from the hierarchical clustering of the response matrix. The easy way to obtain this is by using the function (tree_parms) which gives a default clustering. However, user decide on a specific structure and then input a tree that follows such structure.
my_print	Should information form each ADMM iteration be printed along the way? This prints the dual and primal residuals
invmat	A list of length ncol(y), each containing the C_d part of equation 32 in the paper
gg	penalty terms for the tree structure for lambda_1 and lambda_2 for the ADMM call.

Value

predicted values for the ADMM part beta0: estimated beta_0 coefficients having a size of 1 by ncol(y)

beta: estimated beta coefficients having a matrix ncol(X) by ncol(y)

BETA_hat: estimated beta and theta coefficients having a matrix (ncol(X)+ncol(X) by ncol(Z)) by ncol(y)

theta0: estimated theta_0 coefficients having a matrix ncol(Z) by ncol(y)

theta: estimated theta coefficients having a an array ncol(X) by ncol(Z) by ncol(y) converge: did the algorithm converge?

Y_HAT: predicted response nrow(X) by ncol(y)

`admm_MADMMplasso_cpp` *Fit the ADMM part of model for a given lambda vale*

Description

This function fits a multi-response pliable lasso model over a path of regularization values.

Usage

```
admm_MADMMplasso_cpp(
  beta0,
  theta0,
  beta,
  beta_hat,
  theta,
  rho1,
  X,
  Z,
  max_it,
  W_hat,
  XtY,
  y,
  N,
  e_abs,
  e_rel,
  alpha,
  lambda,
  alph,
  svd_w_tu,
  svd_w_tv,
  svd_w_d,
  C,
  CW,
  gg,
```

```
  my_print = TRUE
)
```

Arguments

beta0	a vector of length ncol(y) of estimated beta_0 coefficients
theta0	matrix of the initial theta_0 coefficients ncol(Z) by ncol(y)
beta	a matrix of the initial beta coefficients ncol(X) by ncol(y)
beta_hat	a matrix of the initial beta and theta coefficients (ncol(X)+ncol(X) by ncol(Z)) by ncol(y)
theta	an array of initial theta coefficients ncol(X) by ncol(Z) by ncol(y)
rho1	the Lagrange variable for the ADMM which is usually included as rho in the MADMMplasso call.
X	n by p matrix of predictors
Z	n by nz matrix of modifying variables. The elements of z may represent quantitative or categorical variables, or a mixture of the two. Categorical variables should be coded by 0-1 dummy variables: for a k-level variable, one can use either k or k-1 dummy variables.
max_it	maximum number of iterations in loop for one lambda during the ADMM optimization. This is usually included in the MADMMplasso call
W_hat	N by (p+(p by nz)) of the main and interaction predictors. This generated internally when MADMMplasso is called or by using the function generate_my_w.
XtY	a matrix formed by multiplying the transpose of X by y.
y	N by D matrix of responses. The X and Z variables are centered in the function. We recommend that X and Z also be standardized before the call
N	nrow(X)
e_abs	absolute error for the ADMM. This is included int the call of MADMMplasso.
e_rel	relative error for the ADMM. This is included int the call of MADMMplasso.
alpha	mixing parameter, usually obtained from the MADMMplasso call. When the goal is to include more interactions, alpha should be very small and vice versa.
lambda	a vector lambda_3 values for the ADMM call with length ncol(y). This is usually calculated in the MADMMplasso call. In our current setting, we use the same the lambda_3 value for all responses.
alph	an overrelaxation parameter in [1, 1.8], usually obtained from the MADMMplasso call.
svd_w_tu	the transpose of the U matrix from the SVD of W_hat
svd_w_tv	the transpose of the V matrix from the SVD of W_hat
svd_w_d	the D matrix from the SVD of W_hat
C	the trained tree
CW	weights for the trained tree The easy way to obtain this is by using the function (tree_parms) which gives a default clustering. However, user decide on a specific structure and then input a tree that follows such structure.

gg	penalty terms for the tree structure for lambda_1 and lambda_2 for the ADMM call.
my_print	Should information form each ADMM iteration be printed along the way? Default TRUE. This prints the dual and primal residuals

Value

predicted values for the ADMM part

`compute_pliable` *Compute the interaction part of the model.*

Description

Compute the interaction part of the model.

Usage

```
compute_pliable(X, Z, theta)
```

Arguments

X	N by p matrix of predictors
Z	N by K matrix of modifying variables. The elements of Z may represent quantitative or categorical variables, or a mixture of the two. Categorical variables should be coded by 0-1 dummy variables: for a k-level variable, one can use either k or k-1 dummy variables.
theta	theta coefficients for a single response ncol(X) by ncol(Z)

Value

a vector of length N of the calculated interaction term for a single response

---

`cv_MADMMplasso` *Carries out cross-validation for a MADMMplasso model over a path of regularization values*

Description

Carries out cross-validation for a MADMMplasso model over a path of regularization values

Usage

```
cv_MADMMplasso(
  fit,
  nfolds,
  X,
  Z,
  y,
  alpha = 0.5,
  lambda = fit$Lambdas,
  max_it = 50000,
  e.abs = 0.001,
  e.rel = 0.001,
  nlambda,
  rho = 5,
  my_print = FALSE,
  alph = 1,
  foldid = NULL,
  pal = cl == 1L,
  gg = c(7, 0.5),
  TT,
  tol = 1e-04,
  cl = 1L,
  legacy = FALSE
)
```

Arguments

fit	object returned by the MADMMplasso function
nfolds	number of cross-validation folds
X	N by p matrix of predictors
Z	N by K matrix of modifying variables. The elements of Z may represent quantitative or categorical variables, or a mixture of the two. Categorical variables should be coded by 0-1 dummy variables: for a k-level variable, one can use either k or k-1 dummy variables.
y	N by D matrix of responses. The X and Z variables are centered in the function. We recommend that X and Z also be standardized before the call
alpha	mixing parameter. When the goal is to include more interactions, alpha should be very small and vice versa.
lambda	user specified lambda_3 values.
max_it	maximum number of iterations in loop for one lambda during the ADMM optimization
e.abs	absolute error for the ADMM
e.rel	relative error for the ADMM
nlambda	number of lambda_3 values desired. Similar to maxgrid but can have a value less than or equal to maxgrid.

rho	the Lagrange variable for the ADMM. This value is updated during the ADMM call based on a certain condition.
my_print	Should information form each ADMM iteration be printed along the way? This prints the dual and primal residuals
alph	an overrelaxation parameter in [1, 1.8]. The implementation is borrowed from Stephen Boyd's MATLAB code
foldid	vector with values in 1:K, indicating folds for K-fold CV. Default NULL
pal	Should the lapply function be applied for an alternative to parallelization.
gg	penalty term for the tree structure. This is a 2×2 matrix values in the first row representing the maximum to the minimum values for lambda_1 and the second row representing the maximum to the minimum values for lambda_2. In the current setting, we set both maximum and the minimum to be same because cross validation is not carried across the lambda_1 and lambda_2. However, setting different values will work during the model fit.
TT	The results from the hierarchical clustering of the response matrix. This should same as the parameter tree used during the MADMMplasso call.
tol	threshold for the non-zero coefficients
cl	The number of CPUs to be used for parallel processing
legacy	If TRUE, use the R version of the algorithm

Value

results containing the CV values

Examples

```
# Train the model
# generate some data
set.seed(1235)
N <- 100
p <- 50
nz <- 4
K <- nz
X <- matrix(rnorm(n = N * p), nrow = N, ncol = p)
mx <- colMeans(X)
sx <- sqrt(apply(X, 2, var))
X <- scale(X, mx, sx)
X <- matrix(as.numeric(X), N, p)
Z <- matrix(rnorm(N * nz), N, nz)
mz <- colMeans(Z)
sz <- sqrt(apply(Z, 2, var))
Z <- scale(Z, mz, sz)
beta_1 <- rep(x = 0, times = p)
beta_2 <- rep(x = 0, times = p)
beta_3 <- rep(x = 0, times = p)
beta_4 <- rep(x = 0, times = p)
beta_5 <- rep(x = 0, times = p)
beta_6 <- rep(x = 0, times = p)
```

```
beta_1[1:5] <- c(2, 2, 2, 2, 2)
beta_2[1:5] <- c(2, 2, 2, 2, 2)
beta_3[6:10] <- c(2, 2, 2, -2, -2)
beta_4[6:10] <- c(2, 2, 2, -2, -2)
beta_5[11:15] <- c(-2, -2, -2, -2, -2)
beta_6[11:15] <- c(-2, -2, -2, -2, -2)

Beta <- cbind(beta_1, beta_2, beta_3, beta_4, beta_5, beta_6)
colnames(Beta) <- 1:6

theta <- array(0, c(p, K, 6))
theta[1, 1, 1] <- 2
theta[3, 2, 1] <- 2
theta[4, 3, 1] <- -2
theta[5, 4, 1] <- -2
theta[1, 1, 2] <- 2
theta[3, 2, 2] <- 2
theta[4, 3, 2] <- -2
theta[5, 4, 2] <- -2
theta[6, 1, 3] <- 2
theta[8, 2, 3] <- 2
theta[9, 3, 3] <- -2
theta[10, 4, 3] <- -2
theta[6, 1, 4] <- 2
theta[8, 2, 4] <- 2
theta[9, 3, 4] <- -2
theta[10, 4, 4] <- -2
theta[11, 1, 5] <- 2
theta[13, 2, 5] <- 2
theta[14, 3, 5] <- -2
theta[15, 4, 5] <- -2
theta[11, 1, 6] <- 2
theta[13, 2, 6] <- 2
theta[14, 3, 6] <- -2
theta[15, 4, 6] <- -2

pliable <- matrix(0, N, 6)
for (e in 1:6) {
  pliable[, e] <- compute_pliable(X, Z, theta[, , e])
}

esd <- diag(6)
e <- MASS::mvrnorm(N, mu = rep(0, 6), Sigma = esd)
y_train <- X %*% Beta + pliable + e
y <- y_train

colnames(y) <- paste0("y", 1:ncol(y))
TT <- tree_parms(y)
plot(TT$h_clust)
gg1 <- matrix(0, 2, 2)
gg1[1, ] <- c(0.02, 0.02)
gg1[2, ] <- c(0.02, 0.02)
```

```
nlambda <- 3
e.abs <- 1E-3
e.rel <- 1E-1
alpha <- 0.2
tol <- 1E-2
fit <- MADMMplasso(
  X, Z, y, alpha = alpha, my_lambda = NULL, lambda_min = 0.001, max_it = 100,
  e.abs = e.abs, e.rel = e.rel, maxgrid = nlambda, nlambda = nlambda, rho = 5,
  tree = TT, my_print = FALSE, alph = 1, gg = gg1, tol = tol, cl = 2L
)
cv_admp <- cv_MADMMplasso(
  fit, nfolds = 5, X, Z, y, alpha = alpha, lambda = fit$Lambdas, max_it = 100,
  e.abs = e.abs, e.rel = e.rel, nlambda, rho = 5, my_print = FALSE, alph = 1,
  foldid = NULL, gg = fit$gg, TT = TT, tol = tol
)
plot(cv_admp)
```

`generate_my_w` *Generate the matrix W as seen in equation 8 for use in the function.*

Description

Generate the matrix W as seen in equation 8 for use in the function.

Usage

```
generate_my_w(X = matrix(), Z = matrix())
```

Arguments

`X`	N by p matrix of predictors
`Z`	N by nz matrix of modifying variables. The elements of z may represent quantitative or categorical variables, or a mixture of the two. Categorical variables should be coded by 0-1 dummy variables: for a k-level variable, one can use either k or k-1 dummy variables.

Value

Generated W matrix nrow(X) by (ncol(X)+ncol(X) by ncol(Z))

MADMMplasso *MADMMplasso: Multi Variate Multi Response ADMM with Interaction Effects*

Description

This system allows one to model a multi-variate, multi-response problem with interaction effects. It combines the usual squared error loss for the multi-response problem with some penalty terms to encourage responses that correlate to form groups and also allow for modeling main and interaction effects that exit within the covariates. The optimization method employed is the Alternating Direction Method of Multipliers (ADMM). The implementation is based on the methodology presented on Quachie Asenso, T., & Zucknick, M. (2023) doi:10.48550/arXiv.2303.11155.

This function fits a multi-response pliable lasso model over a path of regularization values.

Usage

```
MADMMplasso(
  X,
  Z,
  y,
  alpha,
  my_lambda = NULL,
  lambda_min = 0.001,
  max_it = 50000,
  e.abs = 0.001,
  e.rel = 0.001,
  maxgrid,
  nlambda,
  rho = 5,
  my_print = FALSE,
  alph = 1.8,
  tree,
  pal = cl == 1L,
  gg = NULL,
  tol = 1e-04,
  cl = 1L,
  legacy = FALSE
)
```

Arguments

X	N by p matrix of predictors
Z	N by K matrix of modifying variables. The elements of Z may represent quantitative or categorical variables, or a mixture of the two. Categorical variables should be coded by 0-1 dummy variables: for a k-level variable, one can use either k or k-1 dummy variables.

y	N by D matrix of responses. The X and Z variables are centered in the function. We recommend that X and Z also be standardized before the call
alpha	mixing parameter. When the goal is to include more interactions, alpha should be very small and vice versa.
my_lambda	user specified lambda_3 values
lambda_min	the smallest value for lambda_3 , as a fraction of max(lambda_3), the (data derived (lammax)) entry value (i.e. the smallest value for which all coefficients are zero)
max_it	maximum number of iterations in loop for one lambda during the ADMM optimization
e.abs	absolute error for the ADMM
e.rel	relative error for the ADMM
maxgrid	number of lambda_3 values desired
nlambda	number of lambda_3 values desired. Similar to maxgrid but can have a value less than or equal to maxgrid.
rho	the Lagrange variable for the ADMM. This value is updated during the ADMM call based on a certain condition.
my_print	Should information form each ADMM iteration be printed along the way? This prints the dual and primal residuals
alph	an overrelaxation parameter in [1, 1.8]. The implementation is borrowed from Stephen Boyd's MATLAB code
tree	The results from the hierarchical clustering of the response matrix. The easy way to obtain this is by using the function (tree_parms) which gives a default clustering. However, user decide on a specific structure and then input a tree that follows such structure.
pal	Should the lapply function be applied for an alternative to parallelization.
gg	penalty term for the tree structure. This is a 2×2 matrix values in the first row representing the maximum to the minimum values for lambda_1 and the second row representing the maximum to the minimum values for lambda_2. In the current setting, we set both maximum and the minimum to be same because cross validation is not carried across the lambda_1 and lambda_2. However, setting different values will work during the model fit.
tol	threshold for the non-zero coefficients
cl	The number of CPUs to be used for parallel processing
legacy	If TRUE, use the R version of the algorithm

Value

predicted values for the MADMMplasso object with the following components: path: a table containing the summary of the model for each lambda_3.

beta0: a list (length=nlambda) of estimated beta_0 coefficients each having a size of 1 by ncol(y)

beta: a list (length=nlambda) of estimated beta coefficients each having a matrix ncol(X) by ncol(y)

BETA_hat: a list (length=nlambda) of estimated beta and theta coefficients each having a matrix (ncol(X)+ncol(X) by ncol(Z)) by ncol(y)

theta0: a list (length=nlambda) of estimated theta_0 coefficients each having a matrix ncol(Z) by ncol(y)

theta: a list (length=nlambda) of estimated theta coefficients each having a an array ncol(X) by ncol(Z) by ncol(y)

Lambdas: values of lambda_3 used

non_zero: number of nonzero betas for each model in path

LOSS: sum of squared of the error for each model in path

Y_HAT: a list (length=nlambda) of predicted response nrow(X) by ncol(y)

gg: penalty term for the tree structure (lambda_1 and lambda_2) for each lambda_3 nlambda by 2

Author(s)

Maintainer: Waldir Leoncio <w.l.netto@medisin.uio.no> (ORCID)

Authors:

- Theophilus Quachie Asenso <t.q.asenso@medisin.uio.no>
- Manuela Zucknick <Manuela.zucknick@medisin.uio.no>
- Chi Zhang <andreachizhang@yahoo.com>

Examples

```
# Train the model
# generate some data
set.seed(1235)
N <- 100
p <- 50
nz <- 4
K <- nz
X <- matrix(rnorm(n = N * p), nrow = N, ncol = p)
mx <- colMeans(X)
sx <- sqrt(apply(X, 2, var))
X <- scale(X, mx, sx)
X <- matrix(as.numeric(X), N, p)
Z <- matrix(rnorm(N * nz), N, nz)
mz <- colMeans(Z)
sz <- sqrt(apply(Z, 2, var))
Z <- scale(Z, mz, sz)
beta_1 <- rep(x = 0, times = p)
beta_2 <- rep(x = 0, times = p)
beta_3 <- rep(x = 0, times = p)
beta_4 <- rep(x = 0, times = p)
beta_5 <- rep(x = 0, times = p)
beta_6 <- rep(x = 0, times = p)

beta_1[1:5] <- c(2, 2, 2, 2, 2)
beta_2[1:5] <- c(2, 2, 2, 2, 2)
```

```
beta_3[6:10] <- c(2, 2, 2, -2, -2)
beta_4[6:10] <- c(2, 2, 2, -2, -2)
beta_5[11:15] <- c(-2, -2, -2, -2, -2)
beta_6[11:15] <- c(-2, -2, -2, -2, -2)

Beta <- cbind(beta_1, beta_2, beta_3, beta_4, beta_5, beta_6)
colnames(Beta) <- 1:6

theta <- array(0, c(p, K, 6))
theta[1, 1, 1] <- 2
theta[3, 2, 1] <- 2
theta[4, 3, 1] <- -2
theta[5, 4, 1] <- -2
theta[1, 1, 2] <- 2
theta[3, 2, 2] <- 2
theta[4, 3, 2] <- -2
theta[5, 4, 2] <- -2
theta[6, 1, 3] <- 2
theta[8, 2, 3] <- 2
theta[9, 3, 3] <- -2
theta[10, 4, 3] <- -2
theta[6, 1, 4] <- 2
theta[8, 2, 4] <- 2
theta[9, 3, 4] <- -2
theta[10, 4, 4] <- -2
theta[11, 1, 5] <- 2
theta[13, 2, 5] <- 2
theta[14, 3, 5] <- -2
theta[15, 4, 5] <- -2
theta[11, 1, 6] <- 2
theta[13, 2, 6] <- 2
theta[14, 3, 6] <- -2
theta[15, 4, 6] <- -2

pliable <- matrix(0, N, 6)
for (e in 1:6) {
  pliable[, e] <- compute_pliable(X, Z, theta[, , e])
}

esd <- diag(6)
e <- MASS::mvrnorm(N, mu = rep(0, 6), Sigma = esd)
y_train <- X %*% Beta + pliable + e
y <- y_train

colnames(y) <- c(paste0("y", seq_len(ncol(y))))
TT <- tree_parms(y)
plot(TT$h_clust)
gg1 <- matrix(0, 2, 2)
gg1[1, ] <- c(0.02, 0.02)
gg1[2, ] <- c(0.02, 0.02)

nlambda <- 1
e.abs <- 1E-4
```

```
e.rel <- 1E-2
alpha <- 0.2
tol <- 1E-3
fit <- MADMMplasso(
  X, Z, y,
  alpha = alpha, my_lambda = matrix(rep(0.2, ncol(y)), 1),
  lambda_min = 0.001, max_it = 1000, e.abs = e.abs, e.rel = e.rel,
  maxgrid = nlambda, nlambda = nlambda, rho = 5, tree = TT, my_print = FALSE,
  alph = TRUE, gg = gg1, tol = tol, cl = 2L
)
```

`predict.MADMMplasso` *Compute predicted values from a fitted MADMMplasso object. Make predictions from a fitted MADMMplasso model*

Description

Compute predicted values from a MADMMplasso object. Make predictions from a fitted MADMMplasso model

Usage

```
## S3 method for class 'MADMMplasso'
predict(object, X, Z, y, lambda = NULL, ...)
```

Arguments

`object`	object returned from a call to MADMMplasso
`X`	N by p matrix of predictors
`Z`	N by nz matrix of modifying variables. These may be observed or the predictions from a supervised learning algorithm that predicts z from test features x and possibly other features.
`y`	N by D matrix of responses.
`lambda`	values of lambda at which predictions are desired. If NULL (default), the path of lambda values from the fitted model. are used. If lambda is not NULL, the predictions are made at the closest values to lambda in the lambda path from the fitted model
`...`	additional arguments to the generic `predict()` method

Value

predicted values

`sim2` *Simulate data for the model. This is the second simulation data used in the paper*

Description

Simulate data for the model

Usage

```
sim2(p = 500, n = 100, m = 24, nz = 4, rho = 0.4, B.elem = 0.5)
```

Arguments

`p`	column for X which is the main effect
`n`	number of observations
`m`	number of responses
`nz`	number of modifiers
`rho`	values to be used in the covariance matrix when generating X
`B.elem`	the value of the non-zero elements in beta?

Value

The simulated data with the following components: Beta: matrix of actual beta coefficients p by m Theta: a m by p by K array of actual theta coefficients Y: a N by m matrix of response variables X: a N by p matrix of covariates Z: a N by K matrix of modifiers

`tree_parms` *Fit the hierarchical tree structure*

Description

Fit the hierarchical tree structure

Usage

```
tree_parms(y = y, h = 0.7)
```

Arguments

`y`	N by D matrix of response variables
`h`	is the tree cut off

Value

A trained tree with the following components: Tree: the tree matrix stored in 1s and 0s Tw: tree weights associated with the tree matrix. Each weight corresponds to a row in the tree matrix. h_clust: Summary of the hclust call y.colnames: names of the response

Index

admm_MADMMplasso, 2
admm_MADMMplasso_cpp, 4

compute_pliable, 6
cv_MADMMplasso, 6

generate_my_w, 10

MADMMplasso, 11
MADMMplasso-package *(*MADMMplasso*)*, 11

predict.MADMMplasso, 15

sim2, 16

tree_parms, 16