Package 'LUCIDus'

October 5, 2023

Title LUCID with Multi-omics Data

Version 3.0.1

Description LUCID version 3, a major update and enhancement from the original release. LUCID version 3 is more robust, and features more powerful model selection, model visualization, inference based on bootstrap resampling, and implements different analysis strategies for multi-omics data with multiple layers. It also incorporates methods to deal with missingness in multi-omics data.

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.2.3

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

Imports mclust, nnet, boot, jsonlite, networkD3, progress

NeedsCompilation no

Author Qiran Jia [aut, cre] (<https://orcid.org/0000-0002-0790-5967>)

Maintainer Qiran Jia <qiranjia@usc.edu>

R topics documented:

`boot_lucid` *Inference of LUCID model based on bootstrap resampling*

Description

Generate `R` bootstrap replicates of LUCID parameters and derive confidence interval (CI) base on bootstrap. Bootstrap replicates are generated based on nonparameteric resampling, implemented by `ordinary` method of codeboot::boot function. Now only work for LUCID early.

Usage

```
boot_lucid(
  G,
  Z,
  Y,
  lucid_model = c("early", "parallel", "serial"),
  CoG = NULL,
  CoY = NULL,
  model,
  conf = 0.95,
  R = 100
)
```

Arguments

`G`	Exposures, a numeric vector, matrix, or data frame. Categorical variable should be transformed into dummy variables. If a matrix or data frame, rows represent observations and columns correspond to variables.
`Z`	Omics data, a numeric matrix or data frame. Rows correspond to observations and columns correspond to variables.
`Y`	Outcome, a numeric vector. Categorical variable is not allowed. Binary outcome should be coded as 0 and 1.
`lucid_model`	Specifying LUCID model, "early" for early integration, "parallel" for lucid in parallel, "serial" for LUCID in serial.Now only work for LUCID early. If "parallel" or "serial", the function will do nothing.

CoG	Optional, covariates to be adjusted for estimating the latent cluster. A numeric vector, matrix or data frame. Categorical variable should be transformed into dummy variables.
CoY	Optional, covariates to be adjusted for estimating the association between latent cluster and the outcome. A numeric vector, matrix or data frame. Categorical variable should be transformed into dummy variables.
model	A LUCID model fitted by estimate_lucid.
conf	A numeric scalar between 0 and 1 to specify confidence level(s) of the required interval(s).
R	An integer to specify number of bootstrap replicates for LUCID model. If feasible, it is recommended to set R >= 1000.

Value

A list, containing the following components:

beta	effect estimate for each exposure
mu	cluster-specific mean for each omics feature
gamma	effect estiamte for the association btween latent cluster and outcome
bootstrap	The boot object returned by boot:boot

Examples

```
## Not run:
# use simulated data
G <- sim_data$G
Z <- sim_data$Z
Y_normal <- sim_data$Y_normal

# fit lucid model
fit1 <- estimate_lucid(G = G, Z = Z, Y = Y_normal, lucid_model = "early", family = "normal", K = 2,
seed = 1008)

# conduct bootstrap resampling
boot1 <- boot_lucid(G = G, Z = Z, Y = Y_normal, lucid_model = "early",model = fit1, R = 100)

# check distribution for bootstrap replicates of the variable of interest
plot(boot1$bootstrap, 1)

# use 90% CI
boot2 <- boot_lucid(G = G, Z = Z, Y = Y_normal, lucid_model = "early", model = fit1, R = 100, conf = 0.9)

## End(Not run)
```

---

check_K *Check Validity of 'K'*

Description

Checks if the elements in 'K' are integers and greater than or equal to 2.

Usage

```
check_K(K)
```

Arguments

K	A vector of integers.

---

check_na *Check missing patterns in omics data Z*

Description

Check missing patterns in omics data Z

Usage

```
check_na(Z)
```

Arguments

Z	A data matrix representing omics data

Value

1. index:indeces for missing values in omics data
2. indicator_na: missing pattern for each observation
3. impute_flag: - flag to initialize imputation. Only happens when sporadic missing pattern is observed

---

estimate_lucid *Fit LUCID models with one or multiple omics layers*

Description

EM algorithm to estimate LUCID with one or multiple omics layers

Usage

```
estimate_lucid(
  lucid_model = c("early", "parallel", "serial"),
  G,
  Z,
  Y,
  CoG = NULL,
  CoY = NULL,
  K,
  init_omic.data.model = "EEV",
  useY = TRUE,
  tol = 0.001,
```

```
  max_itr = 1000,
  max_tot.itr = 10000,
  Rho_G = 0,
  Rho_Z_Mu = 0,
  Rho_Z_Cov = 0,
  family = c("normal", "binary"),
  seed = 123,
  init_impute = c("mix", "lod"),
  init_par = c("mclust", "random"),
  verbose = FALSE
)
```

Arguments

lucid_model	Specifying LUCID model, "early" for early integration, "parallel" for lucid in parallel, "serial" for lucid in serial
G	an N by P matrix representing exposures
Z	If "early", an N by M matrix; If "parallel", a list, each element is a matrix with N rows; If "serial", a list, each element is a matrix with N rows or a list with two or more matrices with N rows
Y	a length N vector
CoG	an N by V matrix representing covariates to be adjusted for G -> X
CoY	an N by K matrix representing covariates to be adjusted for X -> Y
K	If "early", an integer;If "parallel",an integer vector, same length as Z; If "serial", a list, each element is either an integer or an list of integers, same length as Z
init_omic.data.model	a vector of strings specifies the geometric model of omics data. If NULL, See more in ?mclust::mclustModelNames
useY	logical, if TRUE, EM algorithm fits a supervised LUCID; otherwise unsupervised LUCID.
max_itr	Maximum iterations of the EM algorithm. If the EM algorithm iterates more than max_itr without converging, the EM algorithm is forced to stop.
max_tot.itr	Max number of total iterations for est_lucid function. est_lucid may conduct EM algorithm for multiple times if the algorithm fails to converge.(to be done)
Rho_G	A scalar. This parameter is the LASSO penalty to regularize exposures. If user wants to tune the penalty, use the wrapper function lucid
Rho_Z_Mu	A scalar. This parameter is the LASSO penalty to regularize cluster-specific means for omics data (Z). If user wants to tune the penalty, use the wrapper function lucid
Rho_Z_Cov	A scalar. This parameter is the graphical LASSO penalty to estimate sparse cluster-specific variance-covariance matrices for omics data (Z). If user wants to tune the penalty, use the wrapper function lucid
family	The distribution of the outcome
seed	Random seed to initialize the EM algorithm
init_impute	Method to initialize the imputation of missing values in LUCID. mix will use mclust:imputeData to implement EM Algorithm for Unrestricted General Location Model by the mix package to impute the missing values in omics data; lod will initialize the imputation via replacing missing values by LOD / sqrt(2). LOD is determined by the minimum of each variable in omics data.

init_par	For "early", an interface to initialize EM algorithm, if mclust, initiate the parameters using the mclust package, if random, initiate the parameters by drawing from a uniform distribution; For "parallel", mclust is the default for quick convergence; For "serial", each sub-model follows the above depending on it is a "early" or "parallel"
verbose	A flag indicates whether detailed information for each iteration of EM algorithm is printed in console. Default is FALSE.(to be done)

Value

A list contains the object below:

1. res_Beta: estimation for G->X associations
2. res_Mu_Sigma: estimation for X->Z associations
3. res_Gamma: estimation for X->Y associations
4. loglik: log likelihood of LUCID model
5. inclusion.p: inclusion probability of cluster assignment for each observation
6. K: umber of latent clusters for "early"/list of numbers of latent clusters for "parallel" and "serial"
7. N: number of observations

Examples

```
## Not run:
i <- 1008
set.seed(i)
G <- matrix(rnorm(500), nrow = 100)
Z1 <- matrix(rnorm(1000),nrow = 100)
Z2 <- matrix(rnorm(1000), nrow = 100)
Z3 <- matrix(rnorm(1000), nrow = 100)
Z4 <- matrix(rnorm(1000), nrow = 100)
Z5 <- matrix(rnorm(1000), nrow = 100)
Z <- list(Z1 = Z1, Z2 = Z2, Z3 = Z3, Z4 = Z4, Z5 = Z5)
Y <- rnorm(100)
CoY <- matrix(rnorm(200), nrow = 100)
CoG <- matrix(rnorm(200), nrow = 100)
fit1 <- estimate_lucid(G = G, Z = Z, Y = Y, K = list(2,2,2,2,2),
lucid_model = "serial",
family = "normal",
seed = i,
CoG = CoG, CoY = CoY,
useY = TRUE)

## End(Not run)
```

---

est_lucid *Fit LUCID models with one or multiple omics layers.*

Description

EM algorithm to estimate LUCID with one or multiple omics layers, workhorse function of estimate_lucid. Don't directly call it.

Usage

```
est_lucid(
  lucid_model = c("early", "parallel"),
  G,
  Z,
  Y,
  CoG = NULL,
  CoY = NULL,
  K,
  init_omic.data.model = "EEV",
  useY = TRUE,
  tol = 0.001,
  max_itr = 1000,
  max_tot.itr = 10000,
  Rho_G = 0,
  Rho_Z_Mu = 0,
  Rho_Z_Cov = 0,
  family = c("normal", "binary"),
  seed = 123,
  init_impute = c("mix", "lod"),
  init_par = c("mclust", "random"),
  verbose = FALSE
)
```

Arguments

lucid_model	Specifying LUCID model, "early" for early integration, "parallel" for lucid in parallel
G	an N by P matrix representing exposures
Z	If "early", an N by M matrix; If "parallel", a list, each element is a matrix with N rows
Y	a length N vector
CoG	an N by V matrix representing covariates to be adjusted for G -> X
CoY	an N by K matrix representing covariates to be adjusted for X -> Y
K	If "early", an integer;If "parallel",an integer vector, same length as Z
init_omic.data.model	a vector of strings specifies the geometric model of omics data. If NULL, See more in ?mclust::mclustModelNames
useY	logical, if TRUE, EM algorithm fits a supervised LUCID; otherwise unsupervised LUCID.
max_itr	Maximum iterations of the EM algorithm. If the EM algorithm iterates more than max_itr without converging, the EM algorithm is forced to stop.
max_tot.itr	Max number of total iterations for est_lucid function. est_lucid may conduct EM algorithm for multiple times if the algorithm fails to converge.(to be done)
Rho_G	A scalar. This parameter is the LASSO penalty to regularize exposures. If user wants to tune the penalty, use the wrapper function lucid
Rho_Z_Mu	A scalar. This parameter is the LASSO penalty to regularize cluster-specific means for omics data (Z). If user wants to tune the penalty, use the wrapper function lucid

Rho_Z_Cov	A scalar. This parameter is the graphical LASSO penalty to estimate sparse cluster-specific variance-covariance matrices for omics data (Z). If user wants to tune the penalty, use the wrapper function lucid
family	The distribution of the outcome
seed	Random seed to initialize the EM algorithm
init_impute	Method to initialize the imputation of missing values in LUCID. mix will use mclust:imputeData to implement EM Algorithm for Unrestricted General Location Model by the mix package to impute the missing values in omics data; lod will initialize the imputation via replacing missing values by LOD / sqrt(2). LOD is determined by the minimum of each variable in omics data.
init_par	For "early", an interface to initialize EM algorithm, if mclust, initiate the parameters using the mclust package, if random, initiate the parameters by drawing from a uniform distribution; For "parallel", mclust is the default for quick convergence
verbose	A flag indicates whether detailed information for each iteration of EM algorithm is printed in console. Default is FALSE.(to be done)

Value

A list contains the object below:

1. res_Beta: estimation for G->X associations
2. res_Mu_Sigma: estimation for X->Z associations
3. res_Gamma: estimation for X->Y associations
4. loglik: log likelihood of LUCID model
5. inclusion.p: inclusion probability of cluster assignment for each observation
6. K: number of latent clusters for "early"/list of numbers of latent clusters for "parallel"
7. N: number of observations

Examples

```
## Not run:
i <- 1008
set.seed(i)
G <- matrix(rnorm(500), nrow = 100)
Z1 <- matrix(rnorm(1000),nrow = 100)
Z2 <- matrix(rnorm(1000), nrow = 100)
Z3 <- matrix(rnorm(1000), nrow = 100)
Z <- list(Z1 = Z1, Z2 = Z2, Z2 = Z3)
CoY <- matrix(rnorm(200), nrow = 100)
CoG <- matrix(rnorm(200), nrow = 100)
Y <- rnorm(100)
fit1 <- est_lucid(G = G, Z = Z, Y = Y, K = c(2, 2, 2), CoG = CoG, CoY = CoY,
lucid_model = "parallel",
family = "normal",
init_omic.data.model = "VVV",
seed = i,
init_impute = "mclust",
init_par = "mclust",
useY = TRUE)

## End(Not run)
```

fill_data *Impute missing data by optimizing the likelihood function*

Description

Impute missing data by optimizing the likelihood function

Usage

```
fill_data(obs, mu, sigma, p, index, lucid_model)
```

Arguments

obs	a vector of length M
mu	a matrix of size M x K
sigma	a matrix of size M x M x K
p	a vector of length K
index	a vector of length M, indicating whether a value is missing or not in the raw data

Value

an observation with updated imputed value

f_GtoX *log S(X|G)*

Description

Calculate the log prior inclusion probability for omics layer X_j given the exposures G

Usage

```
f_GtoX(G, Beta_matrix)
```

Arguments

G	an N by P matrix
Beta_matrix	a (P + 1) by (K - 1) matrix

Value

an N by K matrix

f_XtoY *log f(Y|X)*

Description

Calculate the log likelihood of outcome Y given all latent variables X

Usage

```
f_XtoY(Y, Delta, family)
```

Arguments

Y	an N by 1 matrix
Delta	a list with parameters related to outcome
family	a string, either gaussian or binomial

Value

for 2 omics layers, an K1 by K2 by N matrix; for 3 omics layers, an K1 by K2 by K3 by N matrix

f_XtoZ *log f(Z|X)*

Description

calculate the log likelihood for omics layer Z_j given the latent cluster X_j

Usage

```
f_XtoZ(Z, Mu_matrix, Sigma_matrix)
```

Arguments

Z	an N by M matrix
Mu_matrix	an M by K matrix
Sigma_matrix	an M by M by K array

Value

an N by K matrix

gen_ci *generate bootstrp ci (normal, basic and percentile)*

Description

generate bootstrp ci (normal, basic and percentile)

Usage

```
gen_ci(x, conf = 0.95)
```

Arguments

x	an object return by boot function
conf	A numeric scalar between 0 and 1 to specify confidence level(s) of the required interval(s).

Value

a matrix, the first column is t0 statistic from original model

indicator *Indicator Function*

Description

Computes an indicator function.

Usage

```
indicator(x)
```

Arguments

x	A numeric value.

Value

1 if x > 1, otherwise 0.

initialize_Beta *Initialize Beta*

Description

Initializes the Beta parameter for the EM algorithm.

Usage

```
initialize_Beta(K, nG)
```

Arguments

K	A vector specifying the number of clusters for each omics layer.
nG	The number of exposures.

initialize_Delta *Initialize Delta*

Description

Initializes the Delta parameter for the EM algorithm.

Usage

```
initialize_Delta(K, CoY, family = c("gaussian", "binomial"), z, Y)
```

Arguments

K	A vector specifying the number of clusters for each omics layer.
CoY	A data frame for covariates (optional).
family	The distribution family ("gaussian" or "binomial").
z	A list of matrices representing the clustering assignments for each omics layer.
Y	The outcome variable.

initialize_Mu *Initialize Mu*

Description

Initializes the Mu parameter for the EM algorithm.

Usage

```
initialize_Mu(K, nZ)
```

Arguments

K	A vector specifying the number of clusters for each omics layer.
nZ	A vector specifying the number of variables for each omics layer.

initialize_Mu_Sigma *Initialize Mu and Sigma*

Description

Initializes the Mu and Sigma parameters for the EM algorithm.

Usage

```
initialize_Mu_Sigma(K, Z, modelNames, na_pattern)
```

Arguments

K	A vector specifying the number of clusters for each omics layer.
Z	A list of matrices representing the data for each omics layer.
modelNames	A list of model names for each omics layer.
na_pattern	A list of NA patterns for each omics layer.

initialize_Sigma *Initialize Sigma*

Description

Initializes the Sigma parameter for the EM algorithm.

Usage

```
initialize_Sigma(K, nZ)
```

Arguments

K	A vector specifying the number of clusters for each omics layer.
nZ	A vector specifying the number of variables for each omics layer.

Istep_Z *I-step of LUCID*

Description

Impute missing data in Z by maximizing the likelihood given fixed parameters of LUCID

Usage

```
Istep_Z(Z, p, mu, sigma, index, lucid_model)
```

Arguments

Z	an N by P matrix representing the omics data
p	an N by K matrix representing inclusion probability for each latent cluster
mu	an M by K matrix representing cluster-specific means
sigma	an M by M by K array representing cluster-specific covariance
index	an N by M matrix representing missing values in Z

Value

a complete dataset of Z

lastInd *Get the Last 'n' Elements of an Array*

Description

Retrieves the last 'n' elements from a vector or array.

Usage

```
lastInd(x, n)
```

Arguments

x	A vector or array.
n	The number of elements to retrieve.

Value

A vector or array containing the last 'n' elements.

LogSumExp *Log-Sum-Exp Trick*

Description

Computes the log-sum-exp trick for a vector.

Usage

```
LogSumExp(vec)
```

Arguments

vec	A numeric vector.

Value

The result of the log-sum-exp trick.

lucid *Fit a lucid model for integrated analysis on exposure, outcome and multi-omics data, allowing for tuning*

Description

Fit a lucid model for integrated analysis on exposure, outcome and multi-omics data, allowing for tuning

Usage

```
lucid(
  G,
  Z,
  Y,
  CoG = NULL,
  CoY = NULL,
  family = c("normal", "binary"),
  K = 2,
  lucid_model = c("early", "parallel", "serial"),
  Rho_G = 0,
  Rho_Z_Mu = 0,
  Rho_Z_Cov = 0,
  verbose_tune = FALSE,
  ...
)
```

Arguments

G	Exposures, a numeric vector, matrix, or data frame. Categorical variable should be transformed into dummy variables. If a matrix or data frame, rows represent observations and columns correspond to variables.
Z	Omics data, a numeric matrix or data frame. Rows correspond to observations and columns correspond to variables. If "early", an N by M matrix; If "parallel", a list, each element is a matrix with N rows; If "serial", a list, each element is a matrix with N rows or a list with two or more matrices with N rows
Y	Outcome, a numeric vector. Categorical variable is not allowed. Binary outcome should be coded as 0 and 1.
CoG	Optional, covariates to be adjusted for estimating the latent cluster. A numeric vector, matrix or data frame. Categorical variable should be transformed into dummy variables.
CoY	Optional, covariates to be adjusted for estimating the association between latent cluster and the outcome. A numeric vector, matrix or data frame. Categorical variable should be transformed into dummy variables.
family	Distribution of outcome. For continuous outcome, use "normal"; for binary outcome, use "binary". Default is "normal".
K	Number of latent clusters (should be greater or equal than 2). Either an integer or a vector of integer. If K is a vector, model selection on K is performed. If "early", an integer;If "parallel",an integer vector, same length as Z; If "serial", a list, each element is either an integer or an list of integers, same length as Z
lucid_model	Specifying LUCID model, "early" for early integration, "parallel" for lucid in parallel, "serial" for lucid in serial
Rho_G	A scalar or a vector. This parameter is the LASSO penalty to regularize exposures. If it is a vector, lucid will call tune_lucid to conduct model selection and variable selection. User can try penalties from 0 to 1. Work or LUCID early only.
Rho_Z_Mu	A scalar or a vector. This parameter is the LASSO penalty to regularize cluster-specific means for omics data (Z). If it is a vector, lucid will call tune_lucid to conduct model selection and variable selection. User can try penalties from 1 to 100. Work or LUCID early only.
Rho_Z_Cov	A scalar or a vector. This parameter is the graphical LASSO penalty to estimate sparse cluster-specific variance-covariance matrices for omics data (Z). If it is a vector, lucid will call tune_lucid to conduct model selection and variable selection. User can try penalties from 0 to 1. Work or LUCID early only.
verbose_tune	A flag to print details of tuning process.
...	Other parameters passed to estimate_lucid
init_omic.data.model	a vector of strings specifies the geometric model of omics

Value

An optimal lucid model

`plot_lucid` *Visualize LUCID model through a Sankey diagram*

Description

In the Sankey diagram, each node either represents a variable (exposure, omics or outcome) or a latent cluster. Each line represents an association. The color of the node represents variable type, either exposure, omics or outcome. The width of the line represents the effect size of a certain association; the color of the line represents the direction of a certain association. Only work for LUCID early for now.

Usage

```
plot_lucid(
  x,
  G_color = "dimgray",
  X_color = "#eb8c30",
  Z_color = "#2fa4da",
  Y_color = "#afa58e",
  pos_link_color = "#67928b",
  neg_link_color = "#d1e5eb",
  fontsize = 7
)
```

Arguments

`x`	A LUCID model fitted by `estimate_lucid`
`G_color`	Color of node for exposure
`X_color`	Color of node for latent cluster
`Z_color`	Color of node for omics data
`Y_color`	Color of node for outcome
`pos_link_color`	Color of link corresponds to positive association
`neg_link_color`	Color of link corresponds to negative association
`fontsize`	Font size for annotation

Value

A DAG graph created by `sankeyNetwork`

predict_lucid *Predict cluster assignment and outcome based on LUCID model*

Description

Predict cluster assignment and outcome based on LUCID model

Usage

```
predict_lucid(
  model,
  lucid_model = c("early", "parallel", "serial"),
  G,
  Z,
  Y = NULL,
  CoG = NULL,
  CoY = NULL,
  response = TRUE
)
```

Arguments

model	A model fitted and returned by estimate_lucid
lucid_model	Specifying LUCID model, "early" for early integration, "parallel" for lucid in parallel "serial" for lucid in serial.
G	Exposures, a numeric vector, matrix, or data frame. Categorical variable should be transformed into dummy variables. If a matrix or data frame, rows represent observations and columns correspond to variables.
Z	Omics data,if "early", a numeric matrix or data frame. Rows correspond to observations and columns correspond to variables; if "parallel", a list, each element is a matrix with N rows; If "serial", a list, each element is a matrix with N rows or a list with two or more matrices with N rows
Y	Outcome, a numeric vector. Categorical variable is not allowed. Binary outcome should be coded as 0 and 1.
CoG	Optional, covariates to be adjusted for estimating the latent cluster. A numeric vector, matrix or data frame. Categorical variable should be transformed into dummy variables.
CoY	Optional, covariates to be adjusted for estimating the association between latent cluster and the outcome. A numeric vector, matrix or data frame. Categorical variable should be transformed into dummy variables.
response	If TRUE, when predicting binary outcome, the response will be returned. If FALSE, the linear predictor is returned.

Value

A list contains predicted latent cluster, PIP and outcome for each observation

Examples

```
## Not run:
# prepare data
G <- sim_data$G
Z <- sim_data$Z
Y_normal <- sim_data$Y_normal

# fit lucid model
fit1 <- estimate_lucid(G = G, Z = Z, Y = Y_normal, lucid_model = "early", K = 2, family = "normal")

# prediction on training set
pred1 <- predict_lucid(model = fit1, G = G, Z = Z, Y = Y_normal, lucid_model = "early")
pred2 <- predict_lucid(model = fit1, G = G, Z = Z, lucid_model = "early")

## End(Not run)
```

`pred_lucid` *Predict cluster assignment and outcome based on LUCID model or "early" and "parallel" only.*

Description

Sub function for predict_lucid_all(). Don't directly call it. Not including lucid in serial.

Usage

```
pred_lucid(
  model,
  lucid_model = c("early", "parallel"),
  G,
  Z,
  Y = NULL,
  CoG = NULL,
  CoY = NULL,
  response = TRUE
)
```

Arguments

`model`	A model fitted and returned by `estimate_lucid`, "early" and "parallel".
`lucid_model`	Specifying LUCID model, "early" for early integration, "parallel" for lucid in parallel.
`G`	Exposures, a numeric vector, matrix, or data frame. Categorical variable should be transformed into dummy variables. If a matrix or data frame, rows represent observations and columns correspond to variables.
`Z`	Omics data,if "early", a numeric matrix or data frame. Rows correspond to observations and columns correspond to variables; if "parallel", a list, each element is a matrix with N rows.
`Y`	Outcome, a numeric vector. Categorical variable is not allowed. Binary outcome should be coded as 0 and 1.

CoG	Optional, covariates to be adjusted for estimating the latent cluster. A numeric vector, matrix or data frame. Categorical variable should be transformed into dummy variables.
CoY	Optional, covariates to be adjusted for estimating the association between latent cluster and the outcome. A numeric vector, matrix or data frame. Categorical variable should be transformed into dummy variables.
response	If TRUE, when predicting binary outcome, the response will be returned. If FALSE, the linear predictor is returned.

Value

A list contains predicted latent cluster and outcome for each observation

---

print.lucid *Print the output of* est_lucid

Description

Print the output of est_lucid

Usage

```
## S3 method for class 'lucid'
print(x, ...)
```

Arguments

x	An object of LUCID model, returned by est_lucid
...	Other arguments to be passed to print

---

print.sumlucid *Print the output of LUCID in a nicer table*

Description

Print the output of LUCID in a nicer table

Usage

```
## S3 method for class 'sumlucid'
print(x, ...)
```

Arguments

x	An object returned by summary_lucid
...	Other parameters to be passed to print

reorder_lucid *function to reorder all model parameters*

Description

function to reorder all model parameters

Usage

```
reorder_lucid(model)
```

Arguments

model	A model returned by EM_lucid

Value

A LUCID model reordered by effect size of outcome

summary_lucid *Summarize results of LUCID model*

Description

Summarize results of LUCID model

Usage

```
summary_lucid(object, boot.se = NULL)
```

Arguments

object	A LUCID model fitted by estimate_lucid
boot.se	An object returned by boot_lucid, which contains the bootstrap confidence intervals

Examples

```
## Not run:
# use simulated data
G <- sim_data$G
Z <- sim_data$Z
Y_normal <- sim_data$Y_normal

# fit lucid model
fit1 <- estimate_lucid(G = G, Z = Z, Y = Y_normal, lucid_model = "early", family = "normal", K = 2,
seed = 1008)

# conduct bootstrap resampling
boot1 <- boot_lucid(G = G, Z = Z, Y = Y_normal, lucid_model = "early", model = fit1, R = 100)
```

```
# summarize lucid model
summary_lucid(fit1)

# summarize lucid model with bootstrap CIs
summary_lucid(fit1, boot.se = boot1)

## End(Not run)
```

---

tune_lucid *A wrapper function to perform model selection for LUCID*

Description

Given a grid of K and L1 penalties (incluing Rho_G, Rho_Z_mu and Rho_Z_Cov; for LUCID early only), fit LUCID model over all combinations of K and L1 penalties to determine the optimal penalty. Note that the input of the grid of K differs for different LUCID models. i.e. For LUCID Early, K = 3:5; for LUCID in parallel, K = list(2:3, 2:3); for LUCID in serial, K = list(list(2:3,2),2:3)

Usage

```
tune_lucid(
  G,
  Z,
  Y,
  CoG = NULL,
  CoY = NULL,
  family = c("normal", "binary"),
  K,
  lucid_model = c("early", "parallel", "serial"),
  Rho_G = 0,
  Rho_Z_Mu = 0,
  Rho_Z_Cov = 0,
  ...
)
```

Arguments

G	Exposures, a numeric vector, matrix, or data frame. Categorical variable should be transformed into dummy variables. If a matrix or data frame, rows represent observations and columns correspond to variables.
Z	If "early", an N by M matrix; If "parallel", a list, each element is a matrix with N rows; If "serial", a list, each element is a matrix with N rows or a list with two or more matrices with N rows.
Y	Outcome, a numeric vector. Categorical variable is not allowed. Binary outcome should be coded as 0 and 1.
CoG	Optional, covariates to be adjusted for estimating the latent cluster. A numeric vector, matrix or data frame. Categorical variable should be transformed into dummy variables.
CoY	Optional, covariates to be adjusted for estimating the association between latent cluster and the outcome. A numeric vector, matrix or data frame. Categorical variable should be transformed into dummy variables.

family	Distribution of outcome. For continuous outcome, use "normal"; for binary outcome, use "binary". Default is "normal".
K	Number of latent clusters. If "early", an integer;If "parallel",an integer vector, same length as Z; If "serial", a list, each element is either an integer or an list of integers, same length as Z. If K is given as a grid, the input of the grid of K differs for different LUCID models. i.e. For LUCID Early, K = 3:5; for LUCID in parallel, K = list(2:3, 2:3); for LUCID in serial, K = list(list(2:3,2),2:3)
Rho_G	A scalar or a vector. This parameter is the LASSO penalty to regularize exposures. If it is a vector, tune_lucid will conduct model selection and variable selection. User can try penalties from 0 to 1. Work or LUCID early only.
Rho_Z_Mu	A scalar or a vector. This parameter is the LASSO penalty to regularize cluster-specific means for omics data (Z). If it is a vector, tune_lucid will conduct model selection and variable selection. User can try penalties from 1 to 100. Work or LUCID early only.
Rho_Z_Cov	A scalar or a vector. This parameter is the graphical LASSO penalty to estimate sparse cluster-specific variance-covariance matrices for omics data (Z). If it is a vector, tune_lucid will conduct model selection and variable selection. User can try penalties from 0 to 1. Work or LUCID early only.
...	Other parameters passed to est_lucid

Value

A list:

best_model	the best model over different combination of tuning parameters
tune_list	a data frame contains combination of tuning parameters and c orresponding BIC
res_model	a list of LUCID models corresponding to each combination of tuning parameters

Examples

```
## Not run:
# use simulated data
G <- sim_data$G
Z <- sim_data$Z
Y_normal <- sim_data$Y_normal

# find the optimal model over the grid of K
tune_K <- tune_lucid(G = G, Z = Z, Y = Y_normal, lucid_model = "early", useY = FALSE, tol = 1e-3,
seed = 1, K = 2:5)

# tune penalties
tune_Rho_G <- tune_lucid(G = G, Z = Z, Y = Y_normal, lucid_model = "early", useY = FALSE, tol = 1e-3,
seed = 1, K = 2, Rho_G = c(0.1, 0.2, 0.3, 0.4))
tune_Rho_Z_Mu <- tune_lucid(G = G, Z = Z, Y = Y_normal, lucid_model = "early", useY = FALSE, tol = 1e-3,
seed = 1, K = 2, Rho_Z_Mu = c(10, 20, 30, 40))
tune_Rho_Z_Cov <- tune_lucid(G = G, Z = Z, Y = Y_normal, lucid_model = "early", useY = FALSE, tol = 1e-3,
seed = 1, K = 2, Rho_Z_Cov = c(0.1, 0.2, 0.3))

## End(Not run)
```

vec_to_array *Transform Mu to an Array*

Description

Transforms a vector of Mu values to an array based on the number of clusters.

Usage

```
vec_to_array(K, mu)
```

Arguments

K	A vector specifying the number of clusters for each omics layer.
mu	A vector of Mu values.

Value

A multi-dimensional array of Mu values.

Index