Package 'LUCIDus'

July 21, 2025

Title LUCID with Multiple Omics Data

Version 3.0.3

Description An implementation of estimating the Latent Unknown Clusters By Integrating Multi-omics Data (LUCID) model (Peng (2019) <doi:10.1093/bioinformatics/btz667>). LUCID conducts integrated clustering using exposures, omics information (and outcome information as an option). This package implements three different integration strategies for multi-omics data analysis within the LUCID framework: LUCID early integration (the original LUCID model), LUCID in parallel (intermediate integration), and LUCID in serial (late integration). Automated model selection for each LUCID model is available to obtain the optimal number of latent clusters, and an integrated imputation approach is implemented to handle sporadic and list-wise missingness in multi-omics data. Lasso-type regularity for exposure and omics features were added. S3 methods for summary and plotting functions were fixed. Fixed minor bugs.

Depends R (>= 3.6.0)

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.2

LazyData true

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

Imports mclust, nnet, boot, jsonlite, networkD3, progress, stats, utils, glasso, glmnet

NeedsCompilation no

Author Qiran Jia [aut, cre] (ORCID: <https://orcid.org/0000-0002-0790-5967>), Yinqi Zhao [aut] (ORCID: <https://orcid.org/0000-0003-2413-732X>), David Conti [ths] (ORCID: <https://orcid.org/0000-0002-2941-7833>), Jesse Goodrich [ctb] (ORCID: <https://orcid.org/0000-0001-6615-0472>)

Maintainer Qiran Jia <qiranjia@usc.edu>

Repository CRAN

Date/Publication 2024-09-23 20:30:02 UTC

Contents

`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 `boot::boot` function. Now only achieved for LUCID early integration.

Usage

```
boot_lucid(
  G,
  Z,
  Y,
  lucid_model = c("early", "parallel", "serial"),
  CoG = NULL,
  CoY = NULL,
  model,
  conf = 0.95,
  R = 100,
  verbose = 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 for LUCID early integration, 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.
verbose	A flag indicates whether detailed information is printed in console. Default is FALSE.

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

```
# 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)

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

---

check_na *Check missing patterns in one layer of omics data Z*

Description

Check missing patterns in one layer of omics data Z

Usage

```
check_na(Z)
```

Arguments

Z	A data matrix representing one layer of omics data

Value

1. index:indeces for missing values in the 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	Omics data, if "early", an N by M matrix; If "parallel", a list, each element i is a matrix with N rows and P_i features; If "serial", a list, each element i is a matrix with N rows and p_i features or a list with two or more matrices with N rows and a certain number of features
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	Number of latent clusters. If "early", an integer greater or equal to 2; If "parallel",an integer vector, same length as Z, with each element being an interger greater or equal to 2; If "serial", a list, each element is either an integer like that for "early" or an list of integers like that for "parallel", 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.
tol	stopping criterion for the EM algorithm

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 estimate_lucid function. estimate_lucid may conduct EM algorithm for multiple times if the algorithm fails to converge.
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. Now only achieved for LUCID early integration.
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.Now only achieved for LUCID early integration.
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. Now only achieved for LUCID early integration.
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.

Value

A list contains the object below:

1. res_Beta: estimation for G->X associations
2. res_Mu: estimation for the mu of the X->Z associations
3. res_Sigma: estimation for the sigma of the X->Z associations
4. res_Gamma: estimation for X->Y associations
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. var.names: names for the G, Z, Y variables
8. init_omic.data.model: pre-specified geometric model of multi-omics data
9. likelihood: converged LUCID model log likelihood
10. family: the distribution of the outcome

11. select: for LUCID early integration only, indicators of whether each exposure and omics feature is selected
12. useY: whether this LUCID model is supervised
13. Z: multi-omics data
14. init_impute: pre-specified imputation method
15. init_par: pre-specified parameter initialization method
16. Rho: for LUCID early integration only, pre-specified regularity tuning parameter
17. N: number of observations
18. submodel: for LUCID in serial only, storing all the submodels

Examples

```
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)
```

---

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
lucid_model	Specifying LUCID model, "early" for early integration, "parallel" for lucid in parallel, "serial" for lucid in serial

Value

an observation with updated imputed value

---

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

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
lucid_model	Specifying LUCID model, "early" for early integration, "parallel" for lucid in parallel, "serial" for lucid in serial

Value

a complete dataset of Z

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. If "early", an N by M matrix; If "parallel", a list, each element i is a matrix with N rows and P_i features; If "serial", a list, each element i is a matrix with N rows and p_i features or a list with two or more matrices with N rows and a certain number of features 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 to be tuned. For lucid_model = "early", 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. For lucid_model = "parallel",a list with vectors of integers or just integers, same length as Z, if the element itself is a vector, model selection on K is performed; For lucid_model = "serial", a list, each element is either an integer or an list of integers, same length as Z, if the smallest element (integer) itself is a vector, model selection on K is performed
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 for 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 for 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 for LUCID early only.
verbose_tune	A flag to print details of tuning process.
...	Other parameters passed to estimate_lucid

Value

An optimal LUCID model

1. res_Beta: estimation for G->X associations
2. res_Mu: estimation for the mu of the X->Z associations
3. res_Sigma: estimation for the sigma of the X->Z associations
4. res_Gamma: estimation for X->Y associations
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. var.names: names for the G, Z, Y variables
8. init_omic.data.model: pre-specified geometric model of multi-omics data
9. likelihood: converged LUCID model log likelihood
10. family: the distribution of the outcome
11. select: for LUCID early integration only, indicators of whether each exposure and omics feature is selected
12. useY: whether this LUCID model is supervised
13. Z: multi-omics data
14. init_impute: pre-specified imputation method
15. init_par: pre-specified parameter initialization method
16. Rho: for LUCID early integration only, pre-specified regularity tuning parameter
17. N: number of observations
18. submodel: for LUCID in serial only, storing all the submodels

Examples

```
# LUCID early integration
G <- sim_data$G
Z <- sim_data$Z
Y_normal <- sim_data$Y_normal
Y_binary <- sim_data$Y_binary
cov <- sim_data$Covariate

# fit lucid model
fit1 <- lucid(G = G, Z = Z, Y = Y_normal, lucid_model = "early", family = "normal")
fit2 <- lucid(G = G, Z = Z, Y = Y_binary, lucid_model = "early", family = "binary", useY = FALSE)
```

```
# including covariates
fit3 <- lucid(G = G, Z = Z, Y = Y_binary, lucid_model = "early", family = "binary", CoG = cov)
fit4 <- lucid(G = G, Z = Z, Y = Y_binary, lucid_model = "early", family = "binary", CoY = cov)

# tune K
fit5 <- lucid(G = G, Z = Z, Y = Y_binary, lucid_model = "early", family = "binary", K = 2:3)

# variable selection
fit6 <- lucid(G = G, Z = Z, Y = Y_binary, lucid_model = "early",
family = "binary", Rho_G = seq(0.01, 0.02, by = 0.01))

# LUCID in parallel
i <- 1008
set.seed(i)
G <- matrix(rnorm(500), nrow = 100)
Z1 <- matrix(rnorm(1000),nrow = 100)
Z2 <- matrix(rnorm(1000), nrow = 100)
Z <- list(Z1 = Z1, Z2 = Z2)
CoY <- matrix(rnorm(200), nrow = 100)
CoG <- matrix(rnorm(200), nrow = 100)
Y <- rnorm(100)
best_parallel <- lucid(G = G, Z = Z, Y = Y, K = list(2:3,2),
CoG = CoG, CoY = CoY, lucid_model = "parallel",
family = "normal", init_omic.data.model = "VVV",
seed = i, init_impute = "mix", init_par = "mclust",
useY = TRUE)

# LUCID in serial
best_serial <- lucid(G = G, Z = Z, Y = Y, K = list(2:3,2),
CoG = CoG, CoY = CoY, lucid_model = "serial",
family = "normal", init_omic.data.model = "VVV",
seed = i, init_impute = "mix", init_par = "mclust",
useY = TRUE)
```

`plot` *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(x, ...)
```

Arguments

x	A LUCID model fitted by estimate_lucid
...	Additional arguments to specify colors and fontsize

Value

A DAG graph created by sankeyNetwork

Examples

```
# prepare data
G <- sim_data$G
Z <- sim_data$Z
Y_normal <- sim_data$Y_normal
Y_binary <- sim_data$Y_binary
cov <- sim_data$Covariate

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

# change node color
plot(fit1, G_color = "yellow")
plot(fit1, Z_color = "red")

# change link color
plot(fit1, pos_link_color = "red", neg_link_color = "green")
```

---

predict_lucid

Predict cluster assignment and outcome based on LUCID model using new data of G,Z,(Y). If g_computation, predict cluster assignment, omics data, and outcome based on LUCID model using new data of G only This function can also be use to extract X assignment is using training data G,Z,Y as input.

Description

Predict cluster assignment and outcome based on LUCID model using new data of G,Z,(Y). If g_computation, predict cluster assignment, omics data, and outcome based on LUCID model using new data of G only This function can also be use to extract X assignment is using training data G,Z,Y as input.

Usage

```
predict_lucid(
  model,
  lucid_model = c("early", "parallel", "serial"),
```

```
  G,
  Z,
  Y = NULL,
  CoG = NULL,
  CoY = NULL,
  response = TRUE,
  g_computation = FALSE,
  verbose = FALSE
)
```

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", an N by M matrix; If "parallel", a list, each element i is a matrix with N rows and P_i features; If "serial", a list, each element i is a matrix with N rows and p_i features or a list with two or more matrices with N rows and a certain number of features
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.
g_computation	If TRUE, the prediction only uses information on G.
verbose	A flag indicates whether detailed information is printed in console. Default is FALSE.

Value

A list containing the following components:

1. inclusion.p: A list of inclusion probabilities for each sub-model in the LUCID model.
2. pred.x: A list of predicted values for the data matrix G.
3. pred.y: Predicted values for the response variable Y (if response is TRUE).
4. pred.z: Predicted values for the omics variables Z (if g_computation is TRUE).

Examples

```
# 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")
```

---

print.sumlucid_early *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_early'
print(x, ...)
```

Arguments

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

Value

A nice table/several nice tables of the summary of the LUCID model

Examples

```
# 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)
```

```
# print the summary of the lucid model in a table
temp <- summary(fit1)
print(temp)
```

`print.sumlucid_parallel` *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_parallel'
print(x, ...)
```

Arguments

`x`	An object returned by `summary`
`...`	Other parameters to be passed to `print.sumlucid_parallel`

Value

A nice table/several nice tables of the summary of the LUCID model

---

`print.sumlucid_serial` *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_serial'
print(x, ...)
```

Arguments

`x`	An object returned by `summary`
`...`	Other parameters to be passed to `print.sumlucid_serial`

Value

A nice table/several nice tables of the summary of the LUCID model

simulated_HELIX_data *A simulated HELIX dataset for LUCID*

Description

The Human Early-Life Exposome (HELIX) project is multi-center research project that aims to characterize early-life environmental exposures and associate these with omics biomarkers and child health outcomes (Vrijheid, 2014. doi: 10.1289/ehp.1307204). We used a subset of HELIX data from Exposome Data Challenge 2021 (hold by ISGlobal) as an example to illustrate LUCID model.

Usage

```
simulated_HELIX_data
```

Format

A list with 4 matrices corresponding to exposures (G), omics data (Z), outcome (Y) and covariates (CoY), a total of 420 observations

exposure 1 exposures measuring the maternal exposure to utero mercury.

omics 10 methylomics, 10 transcriptomics, 10 miRNA

outcome A continuous outcome as an indicator of metabolic-dysfunciton-associated fatty liver disease (MAFLD)

covariate 3 covariates including fish_preg_ter, child sex, maternal age

sim_data *A simulated dataset for LUCID*

Description

This is an example dataset to illustrate LUCID model. It is simulated by assuming there are 2 latent clusters in the data. We assume the exposures are associated with latent cluster which ultimately affects the PFAS concentration and liver injury in children. The latent clusters are also characterized by differential levels of metabolites.

Usage

```
sim_data
```

Format

A list with 5 matrices corresponding to exposures (G), omics data (Z), a continuous outcome, a binary outcome and 2 covariates (can be used either as CoX or CoY). Each matrice contains 2000 observations.

G 10 exposures

Z 10 metabolites

Y_normal Outcome, PFAS concentration in children

Y_bninary Bianry outcome, liver injury status

Covariates 2 continous covariates, can be treated as either CoX or CoY

X Latent clusters

---

summary.early_lucid *Summarize results of the early LUCID model*

Description

Summarize results of the early LUCID model

Usage

```
## S3 method for class 'early_lucid'
summary(object, ...)
```

Arguments

object	A LUCID model fitted by estimate_lucid
...	Additional argument of boot.se, which is an object returned by boot_lucid

Value

A list, containing the extracted key parameters from the LUCID model that can be used to print the summary table

Examples

```
# 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(fit1)

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

`summary.lucid_parallel` *Summarize results of the parallel LUCID model*

Description

Summarize results of the parallel LUCID model

Usage

```
## S3 method for class 'lucid_parallel'
summary(object, ...)
```

Arguments

`object`	A LUCID model fitted by `estimate_lucid`
`...`	Additional argument of `boot.se`, which is an object returned by `boot_lucid`

Value

A list, containing the extracted key parameters from the LUCID model that can be used to print the summary table

---

`summary.lucid_serial` *Summarize results of the serial LUCID model*

Description

Summarize results of the serial LUCID model

Usage

```
## S3 method for class 'lucid_serial'
summary(object, ...)
```

Arguments

object	A LUCID model fitted by estimate_lucid
...	Additional argument of boot.se, which is an object returned by boot_lucid

Value

A list, containing the extracted key parameters from the LUCID model that can be used to print the summary table

---

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,
  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, if "early", an N by M matrix; If "parallel", a list, each element i is a matrix with N rows and P_i features; If "serial", a list, each element i is a matrix with N rows and p_i features or a list with two or more matrices with N rows and a certain number of features

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",a list, each element is an integer/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)
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, tune_lucid will conduct model selection and variable selection. User can try penalties from 0 to 1. Work for 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 for 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 for LUCID early only.
verbose_tune	A flag to print details of tuning process.
...	Other parameters passed to estimate_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-2,
seed = 1, K = 2:3)

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

## End(Not run)
```

Index