Package 'LUCIDus'

May 7, 2026

Title LUCID with Multiple Omics Data

Version 3.1.0

Description Implements Latent Unknown Clusters By Integrating Multi-omics Data (LUCID; Peng (2019) <doi:10.1093/bioinformatics/btz667>) for integrative clustering with exposures, multi-omics data, and health outcomes. Supports three integration strategies: early, parallel, and serial. Provides model fitting and tuning, lasso-type regularization for exposure and omics feature selection, handling of missing data, including both sporadic and complete-case patterns, prediction, and g-computation for estimating causal effects of exposures, bootstrap inference for uncertainty estimation, and S3 summary and plot methods. For the multi-omics integration framework, see Jia (2024) <https://journal.r-project.org/articles/RJ-2024-012/RJ-2024-012.pdf>. For the missing-data imputation mechanism, see Jia (2024) <doi:10.1093/bioadv/vbae123>.

Depends R (>= 3.6.0)

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.3

LazyData true

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

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

URL https://journal.r-project.org/articles/RJ-2024-012/RJ-2024-012.pdf, https://doi.org/10.1093/bioadv/vbae123

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 2026-03-11 08:10:16 UTC

Contents

`analyze_missing_pattern`

Analyze missing data patterns in detail

Description

Analyze missing data patterns in detail

Usage

```
analyze_missing_pattern(Z)
```

Arguments

Z	Matrix of data to analyze for missing patterns

Value

List containing detailed missing data analysis

---

boot_lucid *Inference of LUCID model based on bootstrap resampling*

Description

Generate R bootstrap replicates of LUCID parameters and derive confidence interval (CI) based on bootstrap. Bootstrap replicates are generated by nonparametric resampling, implemented with the ordinary method of boot::boot. Supports lucid_model = "early", lucid_model = "parallel", and lucid_model = "serial".

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/data frame; for LUCID in parallel, a list of numeric matrices/data frames. 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. Bootstrap inference is implemented for all three model types.

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. If the fitted model uses nonzero penalties, boot_lucid will automatically refit a zero-penalty model as fallback because bootstrap inference is only supported for Rho_G = Rho_Z_Mu = Rho_Z_Cov = 0.
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:

beta	Bootstrap CI table(s) for G-to-X effects. For lucid_model = "parallel", this is a list by omics layer and includes the multinomial intercept plus exposures in G (not CoG).
mu	Bootstrap CI table(s) for cluster-specific means of omics features. For lucid_model = "parallel", this is a list by omics layer.
gamma	Bootstrap CI table for X-to-Y parameters.
stage	For lucid_model = "serial", a list of stage-wise CI tables (each stage contains beta, mu, and gamma for the final stage only).
bootstrap	The boot object returned by boot::boot.

Examples

```
# use simulated data
G <- sim_data$G[1:300, , drop = FALSE]
Z <- sim_data$Z[1:300, , drop = FALSE]
Y_normal <- sim_data$Y_normal[1:300]

# 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 <- suppressWarnings(
  boot_lucid(G = G, Z = Z, Y = Y_normal,
             lucid_model = "early", model = fit1, R = 5)
)
```

```
# Use 90% CI
boot2 <- suppressWarnings(
  boot_lucid(G = G, Z = Z, Y = Y_normal, lucid_model = "early",
             model = fit1, R = 5, conf = 0.9)
)
```

`check_and_stabilize_sigma` *Check matrix condition and stabilize if needed*

Description

Check matrix condition and stabilize if needed

Usage

```
check_and_stabilize_sigma(sigma, threshold = 1e+10, epsilon = 1e-06)
```

Arguments

`sigma`	Covariance matrix to check
`threshold`	Condition number threshold (default 1e10)
`epsilon`	Small constant for regularization (default 1e-6)

Value

Stabilized covariance matrix

`check_convergence` *Check convergence with both absolute and relative criteria*

Description

Check convergence with both absolute and relative criteria

Usage

```
check_convergence(old_val, new_val, abs_tol = 1e-06, rel_tol = 1e-06)
```

Arguments

`old_val`	Previous value
`new_val`	New value
`abs_tol`	Absolute tolerance
`rel_tol`	Relative tolerance

Value

Boolean indicating convergence

`check_imputation_quality` *Check quality of imputed data*

Description

Check quality of imputed data

Usage

```
check_imputation_quality(original, imputed)
```

Arguments

`original`	Original data matrix with missing values
`imputed`	Imputed data matrix

Value

List containing imputation quality metrics and overall assessment

---

`check_na` *Check missing patterns in omics data*

Description

Check missing patterns in omics data

Usage

```
check_na(Z, lucid_model = c("early", "parallel"))
```

Arguments

`Z`	A data matrix or list of matrices for parallel model
`lucid_model`	Model type ("early" or "parallel")

Value

List containing missing data analysis

`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 integer 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 exposure coefficients in the G-to-X model. `CoG` adjustment covariates are included unpenalized. If user wants to tune the penalty, use the wrapper function `lucid`. Penalty tuning is supported for "early" and "parallel". For "serial", only scalar penalty inputs are supported.

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`. Penalty tuning is supported for "early" and "parallel". For "serial", only scalar penalty inputs are supported.

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`. Penalty tuning is supported for "early" and "parallel". For "serial", only scalar penalty inputs are supported.

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 — Logging level for fitting progress. If `FALSE`, concise start/finish status lines are printed. If `TRUE`, detailed iteration-level traces (including log-likelihood updates) are printed.

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: number 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 "early" and "parallel", feature-selection indicators. For "parallel", `select$selectG` is the exposure-wise union across layers (selected in at least one layer) and `select$selectG_layer` stores per-layer exposure selection.
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 "early" and "parallel", pre-specified regularity tuning parameters
17. N: number of observations
18. submodel: for LUCID in serial only, storing all the submodels
19. em_control: EM stopping controls used for fitting (`tol`, `max_itr`, `max_tot.itr`); used by bootstrap refits

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 K x M for parallel model, M x K for early model
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
lucid_model	Specifying LUCID model

Value

an observation with updated imputed values

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 the point estimate from original model

Istep_Z *I-step of LUCID*

Description

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

Usage

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

Arguments

Z	Matrix or list of matrices for omics data
p	Probability matrix/vector or list for parallel model
mu	Mean matrix or list of matrices
sigma	Covariance array or list of arrays
index	Missing value indicators
lucid_model	Model type

Value

Imputed dataset

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 either a matrix with N rows and p_i features, 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 exposure coefficients in the G-to-X model; CoG covariates are not penalized. 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. Penalty tuning is supported for "early" and "parallel". For "serial", only scalar penalty inputs are supported.
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. Penalty tuning is supported for "early" and "parallel". For "serial", only scalar penalty inputs are supported.

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. Penalty tuning is supported for "early" and "parallel". For "serial", only scalar penalty inputs are supported.
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: number 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 "early" and "parallel", feature-selection indicators. For "parallel", select$selectG is the exposure-wise union across layers and select$selectG_layer stores per-layer exposure selection.
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 "early" and "parallel", pre-specified regularity tuning parameters
17. N: number of observations
18. submodel: for LUCID in serial only, storing all the submodels

Examples

```
# LUCID early integration (quick smoke example)
G <- sim_data$G[1:80, , drop = FALSE]
Z <- sim_data$Z[1:80, , drop = FALSE]
Y <- sim_data$Y_normal[1:80]
fit_early <- lucid(
  G = G, Z = Z, Y = Y,
  lucid_model = "early", family = "normal", K = 2,
  max_itr = 30, max_tot.itr = 60, seed = 1008
```

```
)

# LUCID in parallel (two layers)
i <- 1008
set.seed(i)
G <- matrix(rnorm(240), nrow = 80)
Z1 <- matrix(rnorm(320), nrow = 80)
Z2 <- matrix(rnorm(320), nrow = 80)
Z <- list(Z1 = Z1, Z2 = Z2)
CoY <- matrix(rnorm(160), nrow = 80)
CoG <- matrix(rnorm(160), nrow = 80)
Y <- rnorm(80)
fit_parallel <- lucid(
  G = G, Z = Z, Y = Y, K = list(2, 2),
  CoG = CoG, CoY = CoY, lucid_model = "parallel",
  family = "normal", seed = i,
  max_itr = 30, max_tot.itr = 60
)
```

`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 From a Fitted LUCID Model*

Description

Predict cluster assignment and outcome using new data on G, Z, and optional Y. If `g_computation = TRUE`, prediction uses only the G-to-X path from the fitted model and returns counterfactual-style predictions under modified G. This function can also be used to extract latent cluster assignments when using the training data as input.

Usage

```
predict_lucid(
  model,
  lucid_model = c("early", "parallel", "serial"),
  G,
  Z = NULL,
  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, and "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). For g_computation = TRUE, Z can be NULL for "early", "parallel", and "serial".
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 outcomes, class labels (0/1) are returned using a 0.5 threshold. If FALSE, predicted probabilities are returned.
g_computation	If TRUE, prediction uses only information on G. For this mode, supplied Z and Y are ignored for "early", "parallel", and "serial".
verbose	A flag indicates whether detailed information is printed in console. Default is FALSE.

Value

A list containing:

inclusion.p	Posterior inclusion probabilities for latent clusters (a matrix for "early"; a list by layer for "parallel" and "serial").
pred.x	Predicted latent-cluster labels (a numeric vector for "early"; a list by layer for "parallel" and "serial").
pred.y	Predicted outcome values. For binary outcomes, this is class labels when response = TRUE and probabilities when response = FALSE.
pred.z	Predicted omics means under g-computation mode (g_computation = 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")

# g-computation style prediction using only G
pred_g <- predict_lucid(model = fit1, G = G, Z = NULL, g_computation = TRUE, 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

Prints a structured model summary, including model specification, missing-data profile, feature-selection overview, model fit statistics, regularization settings, and detailed parameter estimates. If `boot.se` is provided in `summary()`, bootstrap CI tables are shown for sections (1) Y, (2) Z, and (3) E.

Examples

```
# use simulated data
G <- sim_data$G[1:300, , drop = FALSE]
Z <- sim_data$Z[1:300, , drop = FALSE]
Y_normal <- sim_data$Y_normal[1:300]

# 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 <- suppressWarnings(
```

```
  boot_lucid(G = G, Z = Z, Y = Y_normal, lucid_model = "early", model = fit1, R = 5)
)

# 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

Prints a structured parallel-model summary, including per-layer missing-data profile, overall and per-layer feature-selection overview, model fit statistics, regularization settings, and detailed parameter estimates for Y, Z, and E. If `boot.se` is provided in `summary()`, bootstrap CI tables are shown for sections (1) Y, (2) Z, and (3) E.

`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

---

safe_impute *Safe imputation for edge cases*

Description

Safe imputation for edge cases

Usage

```
safe_impute(Z, method = c("mean", "median", "lod"))
```

Arguments

Z	Matrix with missing values
method	Imputation method ("mean", "median", "lod")

Value

Imputed matrix

---

safe_log_sum_exp *Safe log-sum-exp computation*

Description

Safe log-sum-exp computation

Usage

```
safe_log_sum_exp(x)
```

Arguments

x	Numeric vector

Value

Numerically stable result

safe_normalize *Safe probability normalization*

Description

Safe probability normalization

Usage

```
safe_normalize(x)
```

Arguments

x	Vector of probabilities

Value

Normalized probabilities

safe_solve *Safe matrix inversion with stability checks*

Description

Safe matrix inversion with stability checks

Usage

```
safe_solve(matrix, tol = 1e-06)
```

Arguments

matrix	Matrix to invert
tol	Tolerance for numerical stabilization

Value

Inverted matrix or NULL if inversion fails

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

`summarize_missing_stats` *Summarize missing-data patterns from check_na output*

Description

Summarize missing-data patterns from check_na output

Usage

```
summarize_missing_stats(na_pattern, lucid_model = c("early", "parallel"))
```

Arguments

`na_pattern`	Output list from check_na
`lucid_model`	Model type ("early" or "parallel")

Value

Missing-data summary list

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 boot.se, which can be an object returned by boot_lucid to display bootstrap CIs in print output.

Value

A list containing model information, fit statistics, feature-selection summaries (overall + by layer), detailed parameter estimates (by layer), missing-data summaries (by layer), and optional bootstrap CI tables.

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

summary_lucid *Summarize results of the early LUCID model*

Description

Summarize results of the early LUCID model

Usage

```
summary_lucid(object, ...)

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

Arguments

object	A LUCID model fitted by estimate_lucid
...	Additional argument boot.se, which can be an object returned by boot_lucid to display bootstrap CIs in print output.

Value

A list containing model information, fit statistics, feature-selection summaries, detailed parameter estimates, missing-data summaries, and optional bootstrap CI tables.

Examples

```
# use simulated data
G <- sim_data$G[1:300, , drop = FALSE]
Z <- sim_data$Z[1:300, , drop = FALSE]
Y_normal <- sim_data$Y_normal[1:300]

# 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 <- suppressWarnings(
  boot_lucid(G = G, Z = Z, Y = Y_normal, lucid_model = "early", model = fit1, R = 5)
)

# summarize lucid model
summary(fit1)

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

tune_lucid *Wrapper for LUCID Model and Penalty Tuning*

Description

Fit a grid of LUCID models over candidate numbers of latent clusters K and (optionally) L1 penalties Rho_G, Rho_Z_Mu, and Rho_Z_Cov. The input format for K differs by lucid_model. For "early", use an integer vector (for example, 2:4). For "parallel", use a list of vectors/integers, one per layer (for example, list(2:3, 2:3, 2)). For "serial", use a nested list as required by the serial model.

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 variables should be transformed into dummy variables.
Z	Omics data. If "early", an N by M matrix. If "parallel", a list of matrices (same N). If "serial", a list matching the serial model structure.
Y	Outcome, a numeric vector. Binary outcomes should be coded as 0/1.
CoG	Optional covariates for the G-to-X model.
CoY	Optional covariates for the X-to-Y model.
family	Outcome family: "normal" or "binary".
K	Candidate latent-cluster values in model-specific format.
lucid_model	LUCID model type: "early", "parallel", or "serial".
Rho_G	Scalar or vector penalty for exposure coefficients in the G-to-X model. CoG covariates are included unpenalized. Vector tuning is supported for "early" and "parallel". For "serial", only scalar inputs are supported.
Rho_Z_Mu	Scalar or vector penalty for cluster-specific Z means. Vector tuning is supported for "early" and "parallel". For "serial", only scalar inputs are supported.

Rho_Z_Cov	Scalar or vector penalty for cluster-specific Z covariance matrices. Vector tuning is supported for "early" and "parallel". For "serial", only scalar inputs are supported.
verbose_tune	Logical; print tuning progress if TRUE.
...	Additional arguments passed to estimate_lucid.

Value

A list containing the tuning table, fitted models, and the selected optimal model. Returned element names differ slightly by model:

- "early": best_model, tune_list, res_model
- "parallel"/"serial": model_opt, tune_K, model_list

Examples

```
## Not run:
G <- sim_data$G
Z <- sim_data$Z
Y <- sim_data$Y_normal
tune_early <- tune_lucid(G = G, Z = Z, Y = Y, lucid_model = "early", K = 2:3)
tune_rho <- tune_lucid(
  G = G, Z = Z, Y = Y, lucid_model = "early", K = 2,
  Rho_G = c(0, 0.1), Rho_Z_Mu = c(0, 5), Rho_Z_Cov = c(0, 0.1)
)

## End(Not run)
```

Index