Package 'DLFM'

May 18, 2026

Type Package

Version 0.2.4

Title Distributed Laplace Factor Model

Description Distributed estimation method is based on a Laplace factor model to solve the estimates of load and specific variance. The philosophy of the package is described in Guangbao Guo. (2022). <doi:10.1007/s00180-022-01270-z>.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.2

Imports MASS, LaplacesDemon, matrixcalc, stats

Depends R (>= 3.5.0)

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

LazyData true

BuildManual yes

NeedsCompilation no

Language en-US

Repository CRAN

Author Guangbao Guo [aut, cre],
Siqi Liu [aut]

Maintainer Guangbao Guo <ggb11111111@163.com>

Date/Publication 2026-05-18 02:30:02 UTC

Contents

`Australian` *Australian*

Description

This dataset contains information about credit card applications. All attribute names and values have been changed to meaningless symbols to protect confidentiality. The dataset includes a mix of continuous and categorical attributes, with some missing values.

Usage

```
data(Australian)
```

Format

A data frame with 690 rows and 15 columns representing different features related to credit card applications.

- A1: Categorical - 0, 1 (formerly: a, b)
- A2: Continuous
- A3: Continuous
- A4: Categorical - 1, 2, 3 (formerly: p, g, gg)
- A5: Categorical - 1 to 14 (formerly: ff, d, i, k, j, aa, m, c, w, e, q, r, cc, x)
- A6: Categorical - 1 to 9 (formerly: ff, dd, j, bb, v, n, o, h, z)
- A7: Continuous
- A8: Categorical - 1, 0 (formerly: t, f)
- A9: Categorical - 1, 0 (formerly: t, f)
- A10: Continuous
- A11: Categorical - 1, 0 (formerly: t, f)
- A12: Categorical - 1, 2, 3 (formerly: s, g, p)
- A13: Continuous
- A14: Continuous
- A15: Class attribute - 1, 2 (formerly: +, -)

Examples

```
# Load the dataset
data(Australian)

# Print the first few rows of the dataset
print(head(Australian))
```

---

bankruptcy *Bankruptcy data*

Description

The data set contain the ratio of retained earnings (RE) to total assets, and the ratio of earnings before interests and taxes (EBIT) to total assets of 66 American firms recorded in the form of ratios. Half of the selected firms had filed for bankruptcy.

Usage

```
data(bankruptcy)
```

Format

A data frame with the following variables:

Y The status of the firm: `0` bankruptcy or `1` financially sound;

RE Ratio of retained earnings to total assets;

EBIT Ratio of earnings before interests and taxes to total assets

Examples

```
data(bankruptcy)
```

`Breast` *Breast*

Description

This dataset contains original clinical cases reported by Dr. Wolberg. The data are grouped chronologically, reflecting the time periods when the samples were collected. The dataset includes various attributes related to breast cancer diagnosis.

Usage

```
data(Breast)
```

Format

A data frame with 699 rows and several columns representing different features related to breast cancer diagnosis.

- `Sample_code_number`: Identification number for the sample.
- `Clump_Thickness`: 1-10
- `Uniformity_of_Cell_Size`: 1-10
- `Uniformity_of_Cell_Shape`: 1-10
- `Marginal_Adhesion`: 1-10
- `Single_Epithelial_Cell_Size`: 1-10
- `Bare_Nuclei`: 1-10 (some values may be missing or revised)
- `Bland_Chromatin`: 1-10
- `Normal_Nucleoli`: 1-10
- `Mitoses`: 1-10
- `Class`: 2 (benign) or 4 (malignant)

Examples

```
# Load the dataset
data(Breast)

# Print the first few rows of the dataset
print(head(Breast))
```

`calculate_DAA` *Distance Between Column Spaces of Two Loading Matrices*

Description

Computes a normalized distance between the subspaces spanned by the columns of two factor loading matrices `Ahat` (estimated) and `A1` (true/reference). The metric is based on the projection matrix of `A1`, regularized to avoid singularity, and is scaled by the number of factors `m` and squared number of variables `p^2` to lie in $[0, 1]$.

Usage

```
calculate_DAA(Ahat, A1, m, p)
```

Arguments

`Ahat`	Estimated factor loading matrix of dimension `p x m`, or at least having `m` columns.
`A1`	True or reference factor loading matrix of dimension `p x m` (or `p x m1`, but typically `m` columns).
`m`	Number of factors (structural dimension) used in the scaling factor.
`p`	Number of variables (observed dimensions).

Details

The distance is defined as

$$D(A_{\text{est}}, A_{\text{true}}) = \sqrt{1 - \frac{1}{mp^2} \operatorname{Re}\left(\operatorname{tr}\left(A_{\text{est}}^\top (A_{\text{true}} A_{\text{true}}^\top + \epsilon I)^{-1} A_{\text{true}} A_{\text{true}}^\top A_{\text{est}}\right)\right)},$$

where $\epsilon = 10^{-6}$ is a small ridge parameter to ensure invertibility of $A_{\text{true}} A_{\text{true}}^\top$.

Value

A scalar in $[0, 1]$ representing the discrepancy between the column spaces of `Ahat` and `A1`. A value of 0 indicates identical subspaces; values closer to 1 indicate greater discrepancy.

Examples

```
set.seed(123)
p <- 10; m <- 3
# True loading matrix (orthonormal columns)
A_true <- qr.Q(qr(matrix(rnorm(p * m), p, m)))
# Slightly perturbed estimate
A_est <- A_true + matrix(rnorm(p * m, sd = 0.05), p, m)
A_est <- qr.Q(qr(A_est))  # re-orthogonalize

calculate_DAA(A_est, A_true, m, p)
```

---

concrete *Concrete Slump Test Data*

Description

This dataset contains measurements related to the slump test of concrete, including input variables (concrete ingredients) and output variables (slump, flow, and compressive strength).

Usage

```
concrete
```

Format

A data frame with 103 rows and 10 columns.

- `Cement`: Amount of cement (kg in one M^3 concrete).
- `Slag`: Amount of slag (kg in one M^3 concrete).
- `Fly_ash`: Amount of fly ash (kg in one M^3 concrete).
- `Water`: Amount of water (kg in one M^3 concrete).
- `SP`: Amount of superplasticizer (kg in one M^3 concrete).
- `Coarse_Aggr`: Amount of coarse aggregate (kg in one M^3 concrete).
- `Fine_Aggr`: Amount of fine aggregate (kg in one M^3 concrete).
- `SLUMP`: Slump of the concrete (cm).
- `FLOW`: Flow of the concrete (cm).
- `Compressive_Strength`: 28-day compressive strength of the concrete (MPa).

Examples

```
# Load the dataset
data(concrete)

# Print the first few rows of the dataset
print(head(concrete))
```

`Dfactor.tests` *Distributed Factor Model Testing with Wald, GRS, PY tests and FDR control*

Description

Performs comprehensive factor model testing in distributed environment across multiple nodes, including joint tests (Wald, GRS, PY), individual asset t-tests, and False Discovery Rate control.

Usage

```
Dfactor.tests(ret, fac, n1, K, q.fdr = 0.05)
```

Arguments

`ret`	A T × N matrix representing the excess returns of N assets at T time points.
`fac`	A T × K matrix representing the returns of K factors at T time points.
`n1`	The number of assets allocated to each node
`K`	The number of nodes
`q.fdr`	The significance level for FDR (False Discovery Rate) testing, defaulting to 5%.

Value

A list containing the following components:

`alpha_list`	List of alpha vectors from each node
`tstat_list`	List of t-statistics from each node
`pval_list`	List of p-values from each node
`Wald_list`	List of Wald test statistics from each node
`p_Wald_list`	List of p-values for Wald tests from each node
`GRS_list`	List of GRS test statistics from each node
`p_GRS_list`	List of p-values for GRS tests from each node
`PY_list`	List of Pesaran and Yamagata test statistics from each node
`p_PY_list`	List of p-values for PY tests from each node
`reject_fdr_list`	List of logical vectors indicating significant assets after FDR correction from each node
`power_proxy_list`	List of number of significant assets after FDR correction from each node
`combined_alpha`	Combined alpha vector from all nodes
`combined_pval`	Combined p-value vector from all nodes
`combined_reject_fdr`	Combined FDR rejection vector from all nodes
`total_power_proxy`	Total number of significant assets across all nodes after FDR correction

Examples

```
set.seed(42)
T <- 120
N <- 100  # Larger dataset for distributed testing
K_factors <- 3
fac <- matrix(rnorm(T * K_factors), T, K_factors)
beta <- matrix(rnorm(N * K_factors), N, K_factors)
alpha <- rep(0, N)
alpha[1:10] <- 0.4 / 100  # 10 non-zero alphas
eps <- matrix(rnorm(T * N, sd = 0.02), T, N)
ret <- alpha + fac %*% t(beta) + eps

# Distributed testing with 4 nodes, each handling 25 assets
results <- Dfactor.tests(ret, fac, n1 = 25, K = 4, q.fdr = 0.05)

# View combined results
cat("Total significant assets after FDR:", results$total_power_proxy, "\n")
cat("Combined results across all nodes:\n")
print(summary(results$combined_alpha))
```

DGulPC *Distributed general unilateral loading principal component*

Description

Distributed general unilateral loading principal component

Usage

```
DGulPC(data, m, n1, K)
```

Arguments

data	is a total data set
m	is the number of principal component
n1	is the length of each data subset
K	is the number of nodes

Value

AU1,AU2,DU3,Shat

Examples

```
library(LaplacesDemon)
library(MASS)
n=1000
p=10
m=5
mu=t(matrix(rep(runif(p,0,1000),n),p,n))
mu0=as.matrix(runif(m,0))
sigma0=diag(runif(m,1))
F=matrix(mvrnorm(n,mu0,sigma0),nrow=n)
A=matrix(runif(p*m,-1,1),nrow=p)
lanor <- rlaplace(n*p,0,1)
epsilon=matrix(lanor,nrow=n)
D=diag(t(epsilon)%*%epsilon)
data=mu+F%*%t(A)+epsilon
DGulPC(data,m=3,n1=128,K=2)
```

DIPC *Distributed Incremental Principal Component Analysis (DIPC)*

Description

Apply IPC in a distributed manner across K nodes.

Usage

```
DIPC(data, m, eta, K)
```

Arguments

data	Matrix of input data (n × p).
m	Number of principal components.
eta	Proportion of initial batch to total data within each node.
K	Number of nodes (distributed splits).

Value

List with per-node results and aggregated averages.

Examples

```
library(LaplacesDemon)
library(MASS)
n=1000
p=10
m=5
mu=t(matrix(rep(runif(p,0,1000),n),p,n))
mu0=as.matrix(runif(m,0))
```

```
sigma0=diag(runif(m,1))
F=matrix(mvrnorm(n,mu0,sigma0),nrow=n)
A=matrix(runif(p*m,-1,1),nrow=p)
lanor <- rlaplace(n*p,0,1)
epsilon=matrix(lanor,nrow=n)
D=diag(t(epsilon)%*%epsilon)
data=mu+F%*%t(A)+epsilon
results <- DIPC(data, m, eta=0.8, K=5)
```

DPC *Distributed principal component*

Description

Distributed principal component

Usage

```
DPC(data, m, n1, K)
```

Arguments

data	is a total data set
m	is the number of principal component
n1	is the length of each data subset
K	is the number of nodes

Value

Ahat,Dhat,Sigmahathat

Examples

```
library(LaplacesDemon)
library(MASS)
n=1000
p=10
m=5
mu=t(matrix(rep(runif(p,0,1000),n),p,n))
mu0=as.matrix(runif(m,0))
sigma0=diag(runif(m,1))
F=matrix(mvrnorm(n,mu0,sigma0),nrow=n)
A=matrix(runif(p*m,-1,1),nrow=p)
lanor <- rlaplace(n*p,0,1)
epsilon=matrix(lanor,nrow=n)
D=diag(t(epsilon)%*%epsilon)
data=mu+F%*%t(A)+epsilon
DPC(data,m=3,n1=128,K=2)
```

DPPC *Distributed projection principal component*

Description

Distributed projection principal component

Usage

```
DPPC(data, m, n1, K)
```

Arguments

data	is a total data set
m	is the number of principal component
n1	is the length of each data subset
K	is the number of nodes

Value

Apro,pro,Sigmahathatpro

Examples

```
library(LaplacesDemon)
library(MASS)
n=1000
p=10
m=5
mu=t(matrix(rep(runif(p,0,1000),n),p,n))
mu0=as.matrix(runif(m,0))
sigma0=diag(runif(m,1))
F=matrix(mvrnorm(n,mu0,sigma0),nrow=n)
A=matrix(runif(p*m,-1,1),nrow=p)
lanor <- rlaplace(n*p,0,1)
epsilon=matrix(lanor,nrow=n)
D=diag(t(epsilon)%*%epsilon)
data=mu+F%*%t(A)+epsilon
DPPC(data,m=3,n1=128,K=2)
```

DSAPC *The distributed stochastic approximation principal component for handling online data sets with highly correlated data across multiple nodes.*

Description

The distributed stochastic approximation principal component for handling online data sets with highly correlated data across multiple nodes.

Usage

```
DSAPC(data, m, eta, n1, K)
```

Arguments

data	is a highly correlated online data set
m	is the number of principal component
eta	is the proportion of online data to total data
n1	is the length of each data subset
K	is the number of nodes

Value

Asa, Dsa (lists containing results from each node)

Examples

```
library(LaplacesDemon)
library(MASS)
n=1000
p=10
m=5
mu=t(matrix(rep(runif(p,0,1000),n),p,n))
mu0=as.matrix(runif(m,0))
sigma0=diag(runif(m,1))
F=matrix(mvrnorm(n,mu0,sigma0),nrow=n)
A=matrix(runif(p*m,-1,1),nrow=p)
lanor <- rlaplace(n*p,0,1)
epsilon=matrix(lanor,nrow=n)
D=diag(t(epsilon)%*%epsilon)
data=mu+F%*%t(A)+epsilon
DSAPC(data=data, m=3, eta=0.8, n1=128, K=2)
```

`estimate_AD_from_OSDR` *Estimate Factor Loadings and Uniquenesses from OSDR Output*

Description

Extracts the estimated factor loading matrix A and the diagonal noise covariance matrix D from the output of `online_sir_lfm`. The subspace estimate `B_hat` is truncated or padded to have exactly `m` columns, used as A, and D is obtained by subtracting the factor covariance contribution from the sample covariance matrix, with a lower bound on the diagonal entries to ensure positivity.

Usage

```
estimate_AD_from_OSDR(data, res, m)
```

Arguments

`data`	The original data matrix (n x p) used in the online SIR procedure.
`res`	A list returned by `online_sir_lfm`, containing at least the component `B_hat` (p x K_est matrix).
`m`	The target number of factors (structural dimension). If `B_hat` has fewer columns, the missing directions are filled with zeros; if more, only the first `m` columns are retained.

Value

A list with components:

`A_est`	Estimated factor loading matrix of dimension p x m.
`D_est`	Estimated diagonal noise covariance matrix of dimension p x p, with diagonal entries forced to be at least 0.01.

Examples

```
## Not run:
set.seed(123)
n <- 500; p <- 20; m <- 3
B_true <- qr.Q(qr(matrix(rnorm(p * m), p, m)))
f <- matrix(rnorm(n * m), n, m)
eps <- matrix(rexp(n * p, rate = 1) - 1, n, p) # Laplace noise
X <- f %*% t(B_true) + eps

# Run online SIR
res <- online_sir_lfm(X, K_true = m, method = "gradient", verbose = FALSE)

# Estimate A and D from the result
AD <- estimate_AD_from_OSDR(X, res, m)
print(head(AD$A_est))
print(diag(AD$D_est)[1:5])
```

```
## End(Not run)
```

`factor.tests` *Factor Model Testing with Wald, GRS, PY tests and FDR control*

Description

Performs comprehensive factor model testing including joint tests (Wald, GRS, PY), individual asset t-tests, and False Discovery Rate control.

Usage

```
factor.tests(ret, fac, q.fdr = 0.05)
```

Arguments

`ret`	A T × N matrix representing the excess returns of N assets at T time points.
`fac`	A T × K matrix representing the returns of K factors at T time points.
`q.fdr`	The significance level for FDR (False Discovery Rate) testing, defaulting to 5%.

Value

A list containing the following components:

`alpha`	N-vector of estimated alphas for each asset
`tstat`	N-vector of t-statistics for testing individual alphas
`pval`	N-vector of p-values for individual alpha tests
`Wald`	Wald test statistic for joint alpha significance
`p_Wald`	p-value for Wald test
`GRS`	GRS test statistic (finite-sample F-test)
`p_GRS`	p-value for GRS test
`PY`	Pesaran and Yamagata test statistic
`p_PY`	p-value for PY test
`reject_fdr`	Logical vector indicating which assets have significant alphas after FDR correction
`fdr_p`	Adjusted p-values using Benjamini-Hochberg procedure
`power_proxy`	Number of significant assets after FDR correction

Examples

```
set.seed(42)
T <- 120
N <- 25
K <- 3
fac <- matrix(rnorm(T * K), T, K)
beta <- matrix(rnorm(N * K), N, K)
alpha <- rep(0, N)
alpha[1:3] <- 0.4 / 100  # 3 non-zero alphas
eps <- matrix(rnorm(T * N, sd = 0.02), T, N)
ret <- alpha + fac %*% t(beta) + eps
results <- factor.tests(ret, fac, q.fdr = 0.05)

# View results
cat("Wald test p-value:", results$p_Wald, "\n")
cat("GRS test p-value:", results$p_GRS, "\n")
cat("PY test p-value:", results$p_PY, "\n")
cat("Significant assets after FDR:", results$power_proxy, "\n")
```

FanPC *Apply the FanPC method to the Laplace factor model*

Description

This function performs Factor Analysis via Principal Component (FanPC) on a given data set. It calculates the estimated factor loading matrix (AF), specific variance matrix (DF), and the mean squared errors.

Usage

```
FanPC(data, m)
```

Arguments

data	A matrix of input data.
m	is the number of principal component

Value

AF,DF,SigmahatF

Examples

```
library(LaplacesDemon)
library(MASS)
n=1000
p=10
m=5
```

```
mu=t(matrix(rep(runif(p,0,1000),n),p,n))
mu0=as.matrix(runif(m,0))
sigma0=diag(runif(m,1))
F=matrix(mvrnorm(n,mu0,sigma0),nrow=n)
A=matrix(runif(p*m,-1,1),nrow=p)
lanor <- rlaplace(n*p,0,1)
epsilon=matrix(lanor,nrow=n)
D=diag(t(epsilon)%*%epsilon)
data=mu+F%*%t(A)+epsilon
results <- FanPC(data, m)
print(results)
```

`frobenius.norm` *Compute the Frobenius Norm of a Matrix*

Description

Calculates the Frobenius norm of a matrix, defined as the square root of the sum of squares of all its entries. This is the standard Euclidean norm when the matrix is treated as a vector.

Usage

```
frobenius.norm(M)
```

Arguments

`M`	A numeric matrix.

Value

A non-negative scalar representing the Frobenius norm of `M`.

Examples

```
M <- matrix(1:6, nrow = 2)
frobenius.norm(M)
```

`Ftest` *Apply the Farmtest method to the Laplace factor model*

Description

This function simulates data from a Lapalce factor model and applies the FarmTest for multiple hypothesis testing. It calculates the false discovery rate (FDR) and power of the test.

Usage

```
Ftest(
  data,
  p1,
  alpha = 0.05,
  K = -1,
  alternative = c("two.sided", "less", "greater")
)
```

Arguments

`data`	A matrix or data frame of simulated or observed data from a Laplace factor model.
`p1`	The number or proportion of non-zero hypotheses.
`alpha`	The significance level for controlling the false discovery rate (default: 0.05).
`K`	The number of factors to estimate (default: -1, meaning auto-detect).
`alternative`	The alternative hypothesis: "two.sided", "less", or "greater" (default: "two.sided").

Value

A list containing the following elements:

`FDR`	The false discovery rate, which is the proportion of false positives among all discoveries (rejected hypotheses).
`Power`	The statistical power of the test, which is the probability of correctly rejecting a false null hypothesis.
`PValues`	A vector of p-values associated with each hypothesis test.
`RejectedHypotheses`	The total number of hypotheses that were rejected by the FarmTest.
`reject`	Indices of rejected hypotheses.
`means`	Estimated means.

Examples

```
library(LaplacesDemon)
library(MASS)
n=1000
p=10
m=5
mu=t(matrix(rep(runif(p,0,1000),n),p,n))
mu0=as.matrix(runif(m,0))
sigma0=diag(runif(m,1))
F=matrix(mvrnorm(n,mu0,sigma0),nrow=n)
A=matrix(runif(p*m,-1,1),nrow=p)
lanor <- rlaplace(n*p,0,1)
epsilon=matrix(lanor,nrow=n)
D=diag(t(epsilon)%*%epsilon)
data=mu+F%*%t(A)+epsilon
p1=40
results <- Ftest(data, p1)
print(results$FDR)
print(results$Power)
```

---

generate_asymmetric_laplace

Generate Multivariate Asymmetric Laplace Random Vectors

Description

Generates a matrix of independent draws from a multivariate asymmetric Laplace distribution. Each row is $\epsilon = \alpha W + Z\sqrt{W}$, where $W \sim \text{Exp}(1)$ independent of $Z \sim N_p(0, \Sigma)$. The parameter α controls the asymmetry (skewness) in each coordinate.

Usage

```
generate_asymmetric_laplace(n, p, alpha, Sigma)
```

Arguments

n	Number of observations (rows) to generate.
p	Dimension of each observation (columns).
alpha	A numeric vector of length p giving the asymmetry (shift) parameters for each coordinate. Can be a scalar if p = 1.
Sigma	A $p \times p$ positive definite covariance matrix for the Gaussian component Z.

Value

An n x p matrix where each row is a draw from the multivariate asymmetric Laplace distribution with shift vector alpha and scale matrix Sigma.

Examples

```
set.seed(123)
p <- 3
alpha <- c(0.5, -0.3, 0.2)
Sigma <- diag(p)
X <- generate_asymmetric_laplace(100, p, alpha, Sigma)
pairs(X)
```

`generate_multivariate_laplace` *Generate Multivariate Symmetric Laplace Random Vectors*

Description

Generates a matrix of independent draws from a multivariate symmetric Laplace distribution. Each row is an independent observation of a p-dimensional random vector $\epsilon = Z\sqrt{W}$, where $Z \sim N_p(0, \Sigma_L)$ and $W \sim \text{Exp}(1)$ independent of Z.

Usage

```
generate_multivariate_laplace(n, p, Sigma_L)
```

Arguments

`n`	Number of observations (rows) to generate.
`p`	Dimension of each observation (columns).
`Sigma_L`	A $p \times p$ positive definite covariance matrix for the Gaussian component Z.

Value

An `n` x `p` matrix where each row is a draw from the multivariate symmetric Laplace distribution with scale matrix Σ_L.

Examples

```
set.seed(123)
p <- 3
Sigma_L <- diag(p)
X <- generate_multivariate_laplace(100, p, Sigma_L)
pairs(X)
```

`generate_skew_laplace` *Generate Multivariate Skew Laplace Random Vectors*

Description

Generates a matrix of independent draws from a multivariate skew Laplace distribution with location vector α, scale matrix Σ, and skewness parameter γ. Each observation is constructed as $\epsilon = \alpha + \gamma V + \sqrt{V}Z$, where $V \sim \text{Gamma}((p+1)/2, 1/2)$ and $Z \sim N_p(0, \Sigma)$ independent of V.

Usage

```
generate_skew_laplace(n, p, alpha, gamma, Sigma)
```

Arguments

`n`	Number of observations (rows) to generate.
`p`	Dimension of each observation (columns).
`alpha`	Numeric vector of length `p` giving the location (shift) parameter. Can be a scalar if `p = 1`.
`gamma`	Numeric vector of length `p` giving the skewness parameter for each coordinate. Can be a scalar if `p = 1`.
`Sigma`	A $p \times p$ positive definite covariance matrix for the Gaussian component Z.

Value

An `n` x `p` matrix where each row is a draw from the specified multivariate skew Laplace distribution.

Examples

```
set.seed(123)
p <- 3
alpha <- c(0.1, -0.2, 0.0)
gamma <- c(0.5, 0.3, -0.4)
Sigma <- diag(p)
X <- generate_skew_laplace(100, p, alpha, gamma, Sigma)
pairs(X)
```

GulPC *General unilateral loading principal component*

Description

General unilateral loading principal component

Usage

```
GulPC(data, m)
```

Arguments

data	is a total data set
m	is the number of first layer principal component

Value

AU1,AU2,DU3,SigmaUhat

Examples

```
library(LaplacesDemon)
library(MASS)
n=1000
p=10
m=5
mu=t(matrix(rep(runif(p,0,1000),n),p,n))
mu0=as.matrix(runif(m,0))
sigma0=diag(runif(m,1))
F=matrix(mvrnorm(n,mu0,sigma0),nrow=n)
A=matrix(runif(p*m,-1,1),nrow=p)
lanor <- rlaplace(n*p,0,1)
epsilon=matrix(lanor,nrow=n)
D=diag(t(epsilon)%*%epsilon)
data=mu+F%*%t(A)+epsilon
GulPC(data=data,m=5)
```

Heart *Heart*

Description

This dataset contains information about heart disease diagnosis, including various clinical attributes and the presence of heart disease in patients. The dataset is commonly used for classification tasks to predict the presence of heart disease.

Usage

```
data(Heart)
```

Format

A data frame with multiple rows and 14 columns representing different features related to heart disease diagnosis.

- age: Age in years (integer).
- sex: Sex (1 = male; 0 = female) (categorical).
- cp: Chest pain type (categorical).
- trestbps: Resting blood pressure (in mm Hg on admission to the hospital) (integer).
- chol: Serum cholesterol in mg/dl (integer).
- fbs: Fasting blood sugar > 120 mg/dl (1 = true; 0 = false) (categorical).
- restecg: Resting electrocardiographic results (categorical).
- thalach: Maximum heart rate achieved (integer).
- exang: Exercise-induced angina (1 = yes; 0 = no) (categorical).
- oldpeak: ST depression induced by exercise relative to rest (integer).
- slope: The slope of the peak exercise ST segment (categorical).
- ca: Number of major vessels (0-3) colored by fluoroscopy (integer).
- thal: Thalassemia (3 = normal; 6 = fixed defect; 7 = reversible defect) (categorical).
- num: Diagnosis of heart disease (angiographic disease status) (integer).

Examples

```
# Load the dataset
data(Heart)

# Print the first few rows of the dataset
print(head(Heart))
```

---

ionosphere *ionosphere Data*

Description

This dataset contains radar returns from the ionosphere, collected by a system in Goose Bay, Labrador. The dataset is used for classifying radar returns as 'good' or 'bad' based on the presence of structure in the ionosphere.

Usage

```
data(ionosphere)
```

Format

A data frame with multiple rows and 35 columns representing different features related to radar returns.

- `Attribute1`: Continuous feature.
- `Attribute2`: Continuous feature.
- `Attribute3`: Continuous feature.
- `Attribute4`: Continuous feature.
- `Attribute5`: Continuous feature.
- `Attribute6`: Continuous feature.
- `Attribute7`: Continuous feature.
- `Attribute8`: Continuous feature.
- `Attribute9`: Continuous feature.
- `Attribute10`: Continuous feature.
- `...`: Additional continuous features (up to Attribute34).
- `Class`: Binary classification target ('good' or 'bad').

Examples

```
# Load the dataset
data(ionosphere)

# Print the first few rows of the dataset
print(head(ionosphere))
```

IPC *Incremental principal component method*

Description

The incremental principal component can handle online data sets with highly correlated.

Usage

```
IPC(data, m, eta)
```

Arguments

`data`	is a highly correlated online data set
`m`	is the number of principal component
`eta`	is the proportion of online data to total data

Value

Ai,Di

Examples

```
library(LaplacesDemon)
library(MASS)
n=1000
p=10
m=5
mu=t(matrix(rep(runif(p,0,1000),n),p,n))
mu0=as.matrix(runif(m,0))
sigma0=diag(runif(m,1))
F=matrix(mvrnorm(n,mu0,sigma0),nrow=n)
A=matrix(runif(p*m,-1,1),nrow=p)
lanor <- rlaplace(n*p,0,1)
epsilon=matrix(lanor,nrow=n)
D=diag(t(epsilon)%*%epsilon)
data=mu+F%*%t(A)+epsilon
IPC(data=data,m=3,eta=0.8)
```

Iris *Iris Data*

Description

The Iris dataset is a classic and widely-used dataset in the field of machine learning and statistics. It contains measurements of sepal length, sepal width, petal length, and petal width for three species of iris plants. The dataset is commonly used for classification tasks.

Usage

```
data(Iris)
```

Format

A data frame with 150 rows and 5 columns representing different features of iris plants.

- `Sepal.Length`: Sepal length in centimeters (continuous).
- `Sepal.Width`: Sepal width in centimeters (continuous).
- `Petal.Length`: Petal length in centimeters (continuous).
- `Petal.Width`: Petal width in centimeters (continuous).
- `Species`: Species of iris plant (categorical): Iris Setosa, Iris Versicolor, or Iris Virginica.

Examples

```
# Load the dataset
data(Iris)

# Print the first few rows of the dataset
print(head(Iris))
```

LFM *Generate Laplace factor models*

Description

The function is to generate Laplace factor model data. The function supports various distribution types for generating the data, including: - 'truncated_laplace': Truncated Laplace distribution - 'log_laplace': Univariate Symmetric Log-Laplace distribution - 'Asymmetric Log_Laplace': Log-Laplace distribution - 'Skew-Laplace': Skew-Laplace distribution

Usage

```
LFM(n, p, m, distribution_type)
```

Arguments

n	An integer specifying the sample size.
p	An integer specifying the sample dimensionality or the number of variables.
m	An integer specifying the number of factors in the model.
distribution_type	A character string indicating the type of distribution to use for generating the data.

Value

A list containing the following elements:

data	A numeric matrix of the generated data.
A	A numeric matrix representing the factor loadings.
D	A numeric matrix representing the uniquenesses, which is a diagonal matrix.

Examples

```
n <- 1000
p <- 10
m <- 5
sigma1 <- 1
sigma2 <- matrix(c(1,0.7,0.7,1), 2, 2)
distribution_type <- "Asymmetric Log_Laplace"
results <- LFM(n, p, m, distribution_type)
print(results)
```

`new_energy_vehicle` *New Energy Vehicle (NEV) Purchase Intention Survey Data*

Description

A questionnaire survey on consumers' purchase intention toward new energy vehicles (NEVs) and its influencing factors. The dataset includes (i) household vehicle purchase history, (ii) attitudes toward policy/product/economic/firm factors measured on a 5-point Likert scale, and (iii) demographic information.

Usage

```
new_energy_vehicle
```

Format

A data frame with 520 rows and multiple variables:

household_ice_owned Whether the household has purchased an internal-combustion (fuel) vehicle (single choice).

household_nev_owned Whether the household has purchased a new energy vehicle (single choice).

policy_subsidy_intention Effect of subsidy policies (e.g., toll exemptions, lower purchase price, low-interest loans) on NEV purchase intention (Likert 5-point).

policy_license_intention Effect of license-plate policies (e.g., free registration, road-restriction privileges) on NEV purchase intention (Likert 5-point).

environmental_intention Effect of environmental concerns on NEV purchase intention (Likert 5-point).

infrastructure_intention Effect of charging infrastructure convenience on NEV purchase intention (Likert 5-point).

driving_experience_factor Effect of driving experience (product factor) on NEV purchase intention (Likert 5-point).

battery_performance_factor Effect of battery performance (range, lifespan, capacity, charging efficiency) on NEV purchase intention (Likert 5-point).

safety_factor Effect of safety and technology maturity/reliability on NEV purchase intention (Likert 5-point).

depreciation_cost_factor Effect of depreciation/durability concerns (economic factor) on NEV purchase intention (Likert 5-point).

purchase_cost_factor Effect of purchase price (economic factor) on NEV purchase intention (Likert 5-point).

charging_cost_factor Effect of charging cost (economic factor) on NEV purchase intention (Likert 5-point).

maintenance_cost_factor Effect of maintenance/repair cost (economic factor) on NEV purchase intention (Likert 5-point).

service_factor Effect of firm service (pre-sales and after-sales) on NEV purchase intention (Likert 5-point).

brand_factor Effect of brand (firm factor) on NEV purchase intention (Likert 5-point).

technology_advantage_factor Effect of perceived technological advantages (firm factor) on NEV purchase intention (Likert 5-point).

purchase_intent Stated intention to purchase an NEV (Likert 5-point).

recommend_intent Willingness to recommend NEVs to others (Likert 5-point).

repurchase_intent Willingness to prioritize buying an NEV next time (Likert 5-point).

gender Gender (single choice).

age Age group (single choice).

education Education level (single choice).

occupation Occupation (single choice).

hukou Household registration type (rural/urban; single choice).

household_income Average monthly household income (categorical; single choice).

Details

The Likert scale options are: A = Strongly disagree, B = Disagree, C = Neutral, D = Agree, E = Strongly agree.

Source

Consumer survey dataset on NEV purchase intention and influencing factors.

`online_sir_lfm` *Online Sufficient Dimension Reduction for Laplace Factor Model (LFM)*

Description

Implements an online SIR algorithm tailored for LFM data, using a proxy response constructed from the current subspace estimate and robust updates to handle heavy-tailed noise. The algorithm supports two optimization methods: gradient-based updates and perturbation-based updates.

Usage

```
online_sir_lfm(
  X,
  K_true = NULL,
  K_max = NULL,
  c_robust = 1.345,
  eta = "auto",
  method = "gradient",
  verbose = FALSE
)
```

Arguments

X	A matrix or data stream of size n x p (rows = observations, cols = features). Can be processed row-by-row in streaming setting.
K_true	Optional true dimension (for monitoring). If NULL, will estimate online via BIC-like criterion.
K_max	Maximum candidate dimension for online selection (default = min(10, ncol(X))).
c_robust	Robustness scale for tanh transformation (default = 1.345, approx. 0.95 efficiency for Gaussian).
eta	Learning rate schedule: either a function of t, or "auto" for 1/t.
method	Optimization method: "gradient" for gradient-based updates with learning rate, or "perturbation" for direct eigenvector computation of the moment matrix (default = "gradient").
verbose	Logical; if TRUE, prints progress and estimated K at each step.

Value

A list with:

B_hat	Final estimated basis matrix (p x K_est)
K_est	Estimated structural dimension
B_path	List of B estimates over time (optional, for debugging)
loss	Reconstruction loss trace (optional)
method_used	The optimization method actually used

Examples

```
set.seed(123)
n <- 500; p <- 20; m <- 3
B_true <- qr.Q(qr(matrix(rnorm(p * m), p, m)))
f <- matrix(rnorm(n * m), n, m)
eps <- matrix(rexp(n * p, rate = 1) - 1, n, p) # Asymmetric Laplace-like noise
X <- f %*% t(B_true) + eps

# Using gradient method (default)
out_grad <- online_sir_lfm(X, K_true = m, verbose = TRUE)

# Using perturbation method
out_pert <- online_sir_lfm(X, K_true = m, method = "perturbation", verbose = TRUE)
```

`osdr_lfm` *Online Sufficient Dimension Reduction for Laplace Factor Models (OSDR-LFM)*

Description

Implements an online SIR-based sufficient dimension reduction method tailored for Laplace Factor Models (LFM) with symmetric, asymmetric, or skewed error structures. Supports distributed deployment via local updates and global aggregation.

Usage

```
osdr_lfm(
  X,
  Y = NULL,
  laplace_type = c("symmetric", "asymmetric", "skewed"),
  K_max = NULL,
  H = NULL,
  method_svd = c("gradient", "perturbation"),
  is_distributed = FALSE,
  node_id = 1,
  sync_interval = 50,
  verbose = FALSE
)
```

Arguments

`X`	numeric matrix (n x p), observations in rows.
`Y`	optional numeric vector (n) of proxy responses (e.g., factor scores). If NULL, uses norm of projection as proxy (unsupervised LFM mode).
`laplace_type`	character; one of "symmetric", "asymmetric", or "skewed".
`K_max`	integer; maximum candidate dimension (default = min(10, p)).
`H`	integer; number of slices for SIR (default = max(5, floor(sqrt(n)))).
`method_svd`	character; "perturbation" or "gradient" (default = "gradient").
`is_distributed`	logical; if TRUE, simulate distributed node behavior.
`node_id`	integer; node identifier (only used if is_distributed = TRUE).
`sync_interval`	integer; how often to "aggregate" in distributed mode (ignored if not distributed).
`verbose`	logical; print progress.

Value

list with B_hat (p x K_est), K_est, lambda_trace, and (if distributed) local_B.

Examples

```
set.seed(42)
n <- 600; p <- 30; m <- 4
A <- qr.Q(qr(matrix(rnorm(p * m), p, m)))
F <- matrix(rnorm(n * m), n, m)
eps <- matrix(rexp(n * p) - rexp(n * p), n, p)
X <- F %*% t(A) + eps

out <- osdr_lfm(X, laplace_type = "asymmetric", K_max = 6, verbose = TRUE)
cat("Estimated K:", out$K_est, "\n")
```

PC *Principal component*

Description

Principal component

Usage

```
PC(data, m)
```

Arguments

data	is a total data set
m	is the number of principal component

Value

Ahat, Dhat, Sigmahat

Examples

```
library(LaplacesDemon)
library(MASS)
n=1000
p=10
m=5
mu=t(matrix(rep(runif(p,0,1000),n),p,n))
mu0=as.matrix(runif(m,0))
sigma0=diag(runif(m,1))
F=matrix(mvrnorm(n,mu0,sigma0),nrow=n)
A=matrix(runif(p*m,-1,1),nrow=p)
lanor <- rlaplace(n*p,0,1)
epsilon=matrix(lanor,nrow=n)
D=diag(t(epsilon)%*%epsilon)
data=mu+F%*%t(A)+epsilon
PC(data,m=5)
```

PPC *Projection principal component*

Description

Projection principal component

Usage

```
PPC(data, m)
```

Arguments

data	is a total data set
m	is the number of principal component

Value

Apro, Dpro, Sigmahatpro

Examples

```
library(LaplacesDemon)
library(MASS)
n=1000
p=10
m=5
mu=t(matrix(rep(runif(p,0,1000),n),p,n))
mu0=as.matrix(runif(m,0))
sigma0=diag(runif(m,1))
F=matrix(mvrnorm(n,mu0,sigma0),nrow=n)
A=matrix(runif(p*m,-1,1),nrow=p)
lanor <- rlaplace(n*p,0,1)
epsilon=matrix(lanor,nrow=n)
D=diag(t(epsilon)%*%epsilon)
data=mu+F%*%t(A)+epsilon
PPC(data=data,m=5)
```

protein *Protein Secondary Structure Data*

Description

This dataset contains protein sequences and their corresponding secondary structures, including beta-sheets (E), helices (H), and coils (_).

Usage

```
protein
```

Format

A data frame with multiple rows and columns representing protein sequences and their secondary structures.

- `Sequence`: Amino acid sequence (using 3-letter codes).
- `Structure`: Secondary structure of the protein (E for beta-sheet, H for helix, _ for coil).
- `Parameters`: Additional parameters for neural networks (to be ignored).
- `Biophysical_Constants`: Biophysical constants (to be ignored).

Examples

```
# Load the dataset
data(protein)

# Print the first few rows of the dataset
print(head(protein))
```

`review` *Review*

Description

This dataset contains travel reviews from TripAdvisor.com, covering destinations in 11 categories across East Asia. Each traveler's rating is mapped to a scale from Terrible (0) to Excellent (4), and the average rating for each category per user is provided.

Usage

```
review
```

Format

A data frame with multiple rows and 12 columns.

- `User_ID`: Unique identifier for each user (Categorical).
- `Art_Galleries`: Average user feedback on art galleries.
- `Dance_Clubs`: Average user feedback on dance clubs.
- `Juice_Bars`: Average user feedback on juice bars.
- `Restaurants`: Average user feedback on restaurants.
- `Museums`: Average user feedback on museums.
- `Resorts`: Average user feedback on resorts.

- `Parks_Picnic_Spots`: Average user feedback on parks and picnic spots.
- `Beaches`: Average user feedback on beaches.
- `Theaters`: Average user feedback on theaters.
- `Religious_Institutions`: Average user feedback on religious institutions.

Examples

```
# Load the dataset
data(review)

# Print the first few rows of the dataset
print(head(review))
```

`riboflavin` *Riboflavin Production Data*

Description

This dataset contains measurements of riboflavin (vitamin B2) production by Bacillus subtilis, a Gram-positive bacterium commonly used in industrial fermentation processes. The dataset includes $n = 71$ observations with $p = 4088$ predictors, representing the logarithm of the expression levels of 4088 genes. The response variable is the log-transformed riboflavin production rate.

Usage

```
data(riboflavin)
```

Format

y Log-transformed riboflavin production rate (original name: `q_RIBFLV`). This is a continuous variable indicating the efficiency of riboflavin production by the bacterial strain.

x A matrix of dimension 71×4088 containing the logarithm of the expression levels of 4088 genes. Each column corresponds to a gene, and each row corresponds to an observation (experimental condition or time point).

Examples

```
# Load the riboflavin dataset
data(riboflavin)

# Display the dimensions of the dataset
print(dim(riboflavin$x))
print(length(riboflavin$y))
```

riboflavinv100 *Riboflavin Production Data (Top 100 Genes)*

Description

This dataset is a subset of the riboflavin production data by Bacillus subtilis, containing $n = 71$ observations. It includes the response variable (log-transformed riboflavin production rate) and the 100 genes with the largest empirical variances from the original dataset.

Usage

```
data(riboflavinv100)
```

Format

y Log-transformed riboflavin production rate (original name: `q_RIBFLV`). This is a continuous variable indicating the efficiency of riboflavin production by the bacterial strain.

x A matrix of dimension 71×100 containing the logarithm of the expression levels of the 100 genes with the largest empirical variances.

Examples

```
# Load the riboflavinv100 dataset
data(riboflavinv100)

# Display the dimensions of the dataset
print(dim(riboflavinv100$x))
print(length(riboflavinv100$y))
```

SAPC *The stochastic approximation principal component can handle online data sets with highly correlated.*

Description

The stochastic approximation principal component can handle online data sets with highly correlated.

Usage

```
SAPC(data, m, eta)
```

Arguments

data	is a highly correlated online data set
m	is the number of principal component
eta	is the proportion of online data to total data

Value

Asa,Dsa

Examples

```
library(LaplacesDemon)
library(MASS)
n=1000
p=10
m=5
mu=t(matrix(rep(runif(p,0,1000),n),p,n))
mu0=as.matrix(runif(m,0))
sigma0=diag(runif(m,1))
F=matrix(mvrnorm(n,mu0,sigma0),nrow=n)
A=matrix(runif(p*m,-1,1),nrow=p)
lanor <- rlaplace(n*p,0,1)
epsilon=matrix(lanor,nrow=n)
D=diag(t(epsilon)%*%epsilon)
data=mu+F%*%t(A)+epsilon
SAPC(data=data,m=3,eta=0.8)
```

Sonar *Sonar*

Description

This dataset contains sonar signals bounced off a metal cylinder (mines) and a roughly cylindrical rock. The task is to classify whether the signal is from a mine or a rock based on the sonar signal patterns.

Usage

```
data(Sonar)
```

Format

A data frame with 208 rows and 61 columns representing different features of sonar signals.

- `Attribute1`: Continuous feature representing energy within a frequency band.
- `Attribute2`: Continuous feature representing energy within a frequency band.
- `Attribute3`: Continuous feature representing energy within a frequency band.
- `...`: Additional continuous features (up to Attribute60).
- `Class`: Categorical target variable ('M' for mine, 'R' for rock).

Examples

```
# Load the dataset
data(Sonar)

# Print the first few rows of the dataset
print(head(Sonar))
```

`vehicle` *In Vehicle Coupon Recommendation Data*

Description

This dataset contains information about coupon recommendations made to drivers in a vehicle, including various contextual features and the outcome of whether the coupon was accepted.

Usage

```
vehicle
```

Format

A data frame with multiple rows and 27 columns representing different features related to coupon recommendations.

- `destination`: Driver's destination - No Urgent Place, Home, Work.
- `passanger`: Passengers in the car - Alone, Friend(s), Kid(s), Partner.
- `weather`: Current weather - Sunny, Rainy, Snowy.
- `temperature`: Temperature in Fahrenheit - 55, 80, 30.
- `time`: Time of day - 2PM, 10AM, 6PM, 7AM, 10PM.
- `coupon`: Type of coupon - Restaurant(<$20), Coffee House, Carry out & Take away, Bar, Restaurant($20-$50).
- `expiration`: Coupon expiration - 1d (1 day), 2h (2 hours).
- `gender`: Driver's gender - Female, Male.
- `age`: Driver's age group - 21, 46, 26, 31, 41, 50plus, 36, below21.
- `maritalStatus`: Driver's marital status - Unmarried partner, Single, Married partner, Divorced, Widowed.
- `has_Children`: Whether the driver has children - 1, 0.
- `education`: Driver's education level - Some college - no degree, Bachelors degree, Associates degree, High School Graduate, Graduate degree (Masters or Doctorate), Some High School.
- `occupation`: Driver's occupation - Various categories including Unemployed, Student, etc.
- `income`: Driver's income range - Various ranges such as $37500 - $49999, $62500 - $74999, etc.
- `Bar`: Frequency of bar visits per month - never, less1, 1~3, gt8, nan4~8.

- CoffeeHouse: Frequency of coffeehouse visits per month - never, less1, 4~8, 1~3, gt8, nan.
- CarryAway: Frequency of getting take-away food per month - n4~8, 1~3, gt8, less1, never.
- RestaurantLessThan20: Frequency of visiting restaurants with average expense <$20 per month - 4~8, 1~3, less1, gt8, never.
- Restaurant20To50: Frequency of visiting restaurants with average expense $20-$50 per month - 1~3, less1, never, gt8, 4~8, nan.
- toCoupon_GEQ15min: Driving distance to the coupon location greater than 15 minutes - 0, 1.
- toCoupon_GEQ25min: Driving distance to the coupon location greater than 25 minutes - 0, 1.
- direction_same: Whether the coupon location is in the same direction as the current destination - 0, 1.
- direction_opp: Whether the coupon location is in the opposite direction of the current destination - 1, 0.
- Y: Whether the coupon was accepted - 1, 0.

Examples

```
# Load the dataset
data(vehicle)

# Print the first few rows of the dataset
print(head(vehicle))
```

---

wholesale *Wholesale Customers Data*

Description

This dataset contains the annual spending amounts of wholesale customers on various product categories, along with their channel and region information.

Usage

```
wholesale
```

Format

A data frame with 440 rows and 8 columns.

- FRESH: Annual spending (m.u.) on fresh products.
- MILK: Annual spending (m.u.) on milk products.
- GROCERY: Annual spending (m.u.) on grocery products.
- FROZEN: Annual spending (m.u.) on frozen products.
- DETERGENTS_PAPER: Annual spending (m.u.) on detergents and paper products.
- DELICATESSEN: Annual spending (m.u.) on delicatessen products.
- CHANNEL: Customers' channel - Horeca (Hotel/Restaurant/Café) or Retail channel (Nominal).
- REGION: Customers' region - Lisbon, Oporto or Other (Nominal).

Examples

```
# Load the dataset
data(wholesale)
```

`Wine` *Wine Data*

Description

The Wine dataset contains the results of a chemical analysis of wines grown in the same region in Italy but derived from three different cultivars. The analysis determined the quantities of 13 constituents found in each of the three types of wines. This dataset is commonly used for classification tasks to determine the origin of wines based on their chemical properties.

Usage

```
data(Wine)
```

Format

A data frame with 178 rows and 14 columns representing different features of wines.

- `Class`: Categorical target variable indicating the type of wine (1, 2, or 3).
- `Alcohol`: Continuous feature representing the alcohol content.
- `Malic_acid`: Continuous feature representing the malic acid content.
- `Ash`: Continuous feature representing the ash content.
- `Alcalinity_of_ash`: Continuous feature representing the alcalinity of ash.
- `Magnesium`: Integer feature representing the magnesium content.
- `Total_phenols`: Continuous feature representing the total phenols content.
- `Flavanoids`: Continuous feature representing the flavanoids content.
- `Nonflavanoid_phenols`: Continuous feature representing the nonflavanoid phenols content.
- `Proanthocyanins`: Continuous feature representing the proanthocyanins content.
- `Color_intensity`: Continuous feature representing the color intensity.
- `Hue`: Continuous feature representing the hue.
- `OD280_OD315_of_diluted_wines`: Continuous feature representing the OD280/OD315 of diluted wines.
- `Proline`: Continuous feature representing the proline content.

Examples

```
# Load the dataset
data(Wine)

# Print the first few rows of the dataset
print(head(Wine))
```

yacht_hydrodynamics *Yacht Hydrodynamics Data*

Description

This dataset contains the hydrodynamic characteristics of sailing yachts, including design parameters and performance metrics.

Usage

```
yacht_hydrodynamics
```

Format

A data frame with 308 rows and 7 columns.

- `Residuary Resistance`: Residuary resistance per unit weight of displacement (performance metric).
- `Longitudinal Position of Center of Buoyancy`: Longitudinal position of the center of buoyancy.
- `Prismatic Coefficient`: Prismatic coefficient.
- `Length-Displacement Ratio`: Length-displacement ratio.
- `Beam-Draft Ratio`: Beam-draft ratio.
- `Length-Beam Ratio`: Length-beam ratio.
- `Froude Number`: Froude number.

Examples

```
# Load the dataset
data(yacht_hydrodynamics)

# Print the first few rows of the dataset
print(head(yacht_hydrodynamics))
```

Index