Package 'jointMeanCov'

October 13, 2022

Type Package

Title Joint Mean and Covariance Estimation for Matrix-Variate Data

Version 0.1.0

Maintainer Michael Hornstein <mdhornstein@gmail.com>

Description Jointly estimates two-group means and covariances
for matrix-variate data and calculates test statistics.
This package implements the algorithms defined in
Hornstein, Fan, Shedden, and Zhou (2018)
<doi:10.1080/01621459.2018.1429275>.

License GPL-2

Encoding UTF-8

LazyData true

Imports glasso, graphics, stats

Suggests knitr, rmarkdown, testthat

RoxygenNote 6.1.1

NeedsCompilation no

Author Michael Hornstein [aut, cre],
Roger Fan [aut],
Kerby Shedden [aut],
Shuheng Zhou [aut]

Repository CRAN

Date/Publication 2019-05-04 07:40:02 UTC

R topics documented:

`centerDataGLSModelSelection`
Center Each Column By Subtracting Group or Global GLS Mean

Description

This function takes a data matrix, an inverse row covariance matrix, group indices (i.e. row indices for membership in groups one and two), and a subset of column indices indicating which columns should be group centered. It returns a centered data matrix. For each group centered column, the two group means are estimated using GLS; then the group one mean is subtracted from entries in group one, and the group two mean is subtracted from entries in group two. For each globally centered column, a single global mean is estimated using GLS and subtracted from each entry in the column. In addition to returning the centered data matrix, this function also returns the means estimated using GLS.

Usage

```
centerDataGLSModelSelection(X, B.inv, group.one.indices, group.two.indices,
  group.cen.indices)
```

Arguments

`X`	a data matrix.
`B.inv`	an inverse row covariance matrix used in GLS
`group.one.indices`	indices of observations in group one.
`group.two.indices`	indices of observations in group two.
`group.cen.indices`	indices of columns to be group centered

Details

Example

```
n <- 4
m <- 3
X <- matrix(1:12, nrow=n, ncol=m)
# Group center the first two columns, globally center
# the last column.
out <- centerDataGLSModelSelection(
  X, B.inv=diag(n), group.one.indices=1:2,
  group.two.indices=3:4,
  group.cen.indices=1:2)
# Display the centered data matrix
print(out$X.cen)
```

Value

Returns a centered data matrix of the same dimensions as the original data matrix.

`X.cen` Centered data matrix.

`group.means.gls`
Group means estimated using GLS; if all columns are globally centered, then `NULL`.

`global.means.gls`
Global means estimated using GLS; if all columns are group centered, then `NULL`.

---

centerDataTwoGroupsByIndices

Center Each Column by Subtracting Group Means

Description

This function takes a data matrix and returns a centered data matrix. For each column, centering is performed by subtracting the corresponding group mean from each entry (i.e. for entries in group one, the group one mean is subtracted, and for entries in group two, the group two mean is subtracted).

Usage

```
centerDataTwoGroupsByIndices(X, group.one.indices, group.two.indices)
```

Arguments

`X` a data matrix.

`group.one.indices`
indices of observations in group one.

`group.two.indices`
indices of observations in group two.

Details

Example

```
X <- matrix(1:12, nrow=4, ncol=3)
X.cen <- centerDataTwoGroupsByIndices(
  X, group.one.indices=1:2, group.two.indices=3:4)
```

Value

Returns a centered data matrix of the same dimensions as the original data matrix.

`centerDataTwoGroupsByModelSelection`
Center Each Column Based on Model Selection

Description

This function takes a data matrix and returns a centered data matrix. For columns with indices in `within.group.indices`, centering is performed by subtracting the corresponding group mean from each entry (i.e. for entries in group one, the group one mean is subtracted, and for entries in group two, the group two mean is subtracted). For other columns, global centering is performed (i.e. subtracting the column mean from each entry).

Usage

```
centerDataTwoGroupsByModelSelection(X, group.one.indices,
  group.two.indices, within.group.indices)
```

Arguments

`X`	a data matrix.
`group.one.indices`	indices of observations in group one.
`group.two.indices`	indices of observations in group two.
`within.group.indices`	indices of columns on which to perform group centering.

Details

Example

```
X <- matrix(1:12, nrow=4, ncol=3)
# Group center the first two columns, globally center
# the third column.
X.cen <- centerDataTwoGroupsByModelSelection(
  X, group.one.indices=1:2, group.two.indices=3:4,
  within.group.indices=1:2)
```

Value

Returns a centered data matrix of the same dimensions as the original data matrix.

`GeminiB` *Estimate Row-Row Covariance Structure Using Gemini*

Description

GeminiB estimates the row-row covariance, inverse covariance, correlation, and inverse correlation matrices using Gemini. For identifiability, the covariance factors A and B are scaled so that A has trace m, where m is the number of columns of X, A is the column-column covariance matrix, and B is the row-row covariance matrix.

Usage

```
GeminiB(X, rowpen, penalize.diagonal = FALSE)
```

Arguments

`X`	Data matrix, of dimensions n by m.
`rowpen`	Glasso penalty parameter.
`penalize.diagonal`	Logical value indicating whether to penalize the off-diagonal entries of the correlation matrix. Default is FALSE.

Value

`corr.B.hat`	estimated correlation matrix.
`corr.B.hat.inv`	estimated inverse correlation matrix.
`B.hat`	estimated covariance matrix.
`B.hat.inv`	estimated inverse covariance matrix.

Examples

```
n1 <- 5
n2 <- 5
n <- n1 + n2
m <- 20
X <- matrix(rnorm(n * m), nrow=n, ncol=m)
rowpen <- sqrt(log(m) / n)
out <- GeminiB(X, rowpen, penalize.diagonal=FALSE)
# Display the estimated correlation matrix rounded to two
# decimal places.
print(round(out$corr.B.hat, 2))
```

GeminiBPath *Estimate Row-Row Covariance Using Gemini for a Sequence of Penalties*

Description

GeminiBPath estimates the row-row covariance, inverse covariance, correlation, and inverse correlation matrices using Gemini with a sequence of penalty parameters. For identifiability, the covariance factors A and B are scaled so that A has trace m, where m is the number of columns of X, A is the column-column covariance matrix, and B is the row-row covariance matrix.

Usage

```
GeminiBPath(X, rowpen.list, penalize.diagonal = FALSE)
```

Arguments

`X`	Data matrix, of dimensions n by m.
`rowpen.list`	Vector of penalty parameters, should be increasing (analogous to the `glassopath` function of the `glasso` package).
`penalize.diagonal`	Logical indicating whether to penalize the off-diagonal entries of the correlation matrix. Default is FALSE.

Value

`corr.B.hat`	array of estimated correlation matrices, of dimension (nrow(X), nrow(X), length(rowpen.list)).
`corr.B.hat.inv`	array of estimated inverse correlation matrices, of dimension (nrow(X), nrow(X), length(rowpen.list)).
`B.hat`	array of estimated covariance matrices, of dimension (nrow(X), nrow(X), length(rowpen.list)).
`B.hat.inv`	array of estimated inverse covariance matrices, of dimension (nrow(X), nrow(X), length(rowpen.list)).

Examples

```
# Generate a data matrix.
n1 <- 5
n2 <- 5
n <- n1 + n2
m <- 20
X <- matrix(rnorm(n * m), nrow=n, ncol=m)

# Apply GeminiBPath for a sequence of penalty parameters.
rowpen.list <- sqrt(log(m) / n) * c(1, 0.5, 0.1)
out <- GeminiBPath(X, rowpen.list, penalize.diagonal=FALSE)

# Display the estimated correlation matrix corresponding
```

```
# to penalty 0.1, rounded to two decimal places.
print(round(out$corr.B.hat[, , 3], 2))
```

GLSMeans *Generalized Least Squares*

Description

This function applies generalized least squares to estimate the unknown parameters of a linear model X = D beta + E, where X has dimension n by m, D has dimension n by k, and beta has dimension k by m.

Usage

```
GLSMeans(X, D, B.inv)
```

Arguments

X	data matrix.
D	design matrix.
B.inv	inverse covariance matrix.

Details

Example

```
X <- matrix(1:12, nrow=4, ncol=3)
D <- twoGroupDesignMatrix(1:2, 3:4)
B.inv <- diag(4)
beta.hat <- GLSMeans(X, D, B.inv)
```

Value

Returns the estimated parameters of the linear model, a matrix of dimensions k by m, where k is the number of columns of D, and m is the number of columns of X.

`jointMeanCovGroupCen` *Estimate Mean and Row-Row Correlation Matrix Using Group Centering*

Description

This function implements Algorithm 1 from Hornstein, Fan, Shedden, and Zhou (2018), doi: 10.1080/01621459.2018.142927
Given an n by m data matrix, with a vector of indices denoting group membership,this function estimates the row-row inverse covariance matrix after a preliminary group centering step, then uses the estimated inverse covariance estimate to perform GLS mean estimation. The function also returns test statistics comparing the group means for each column, with standard errors accounting for row-row correlation.

Usage

```
jointMeanCovGroupCen(X, group.one.indices, rowpen, B.inv = NULL)
```

Arguments

`X`	Data matrix.
`group.one.indices`	Vector of indices denoting rows in group one.
`rowpen`	Glasso penalty for estimating B, the row correlation matrix.
`B.inv`	Optional row-row covariance matrix to be used in GLS. If this argument is passed, then it is used instead of estimating the inverse row-row covariance.

Value

`B.hat.inv`	Estimated row-row inverse covariance matrix. For identifiability, A and B are scaled so that A has trace m, where m is the number of columns of X.
`corr.B.hat.inv`	Estimated row-row inverse correlation matrix.
`gls.group.means`	Matrix with two rows and m columns, where m is the number of columns of X. Entry (i, j) contains the estimated mean of the jth column for an individual in group i, with i = 1,2, and j = 1, ..., m.
`gamma.hat`	Estimated group mean differences.
`test.stats`	Vector of test statistics of length m.
`p.vals`	Vector of two-sided p-values, calculated using the standard normal distribution.
`p.vals.adjusted`	Vector of p-values, adjusted using the Benjamini-Hochberg fdr adjustment.

Examples

```
# Define sample sizes
n1 <- 5
n2 <- 5
n <- n1 + n2
m <- 200

# Generate data with row and column covariance
# matrices each autorogressive of order one with
# parameter 0.2.  The mean is defined so the first
# three columns have true differences in group means
# equal to four.
Z <- matrix(rnorm(m * n), nrow=n, ncol=m)
A <- outer(1:m, 1:m, function(i, j) 0.2^abs(i - j))
B <- outer(1:n, 1:n, function(i, j) 0.2^abs(i - j))
M <- matrix(0, nrow=nrow(Z), ncol=ncol(Z))
group.one.indices <- 1:5
group.two.indices <- 6:10
M[group.one.indices, 1:3] <- 2
M[group.two.indices, 1:3] <- -2
X <- t(chol(B)) %*% Z %*% chol(A) + M

# Apply Algorithm 1 (jointMeanCovGroupCen) and
# plot the test statistics.
rowpen <- sqrt(log(m) / n)
out <- jointMeanCovGroupCen(X, group.one.indices, rowpen)
plot(out)
summary(out)
```

`jointMeanCovModSelCen` *Estimate Mean and Row-Row Correlation Matrix Using Model Selection*

Description

This function implements Algorithm 2 from Hornstein, Fan, Shedden, and Zhou (2018), doi: 10.1080/01621459.2018.142927
Given an n by m data matrix, with a vector of indices denoting group membership, this function estimates mean and covariance structure as follows. 1. Run Algorithm 1 (`jointMeanCovGroupCen`). 2. Use a threshold to select genes with the largest mean differences. 3. Group center the genes with mean differences above the threshold, and globally center the remaining genes. Use the centered data matrix to calculate a Gram matrix as input to Gemini. 4. Use Gemini to estimate the inverse row covariance matrix, and use the inverse row covariance matrix with GLS to estimate group means. 5. Calculate test statistics comparing group means for each column.

Usage

```
jointMeanCovModSelCen(X, group.one.indices, rowpen, B.inv = NULL,
  rowpen.ModSel = NULL, thresh = NULL)
```

Arguments

X	Data matrix.
group.one.indices	Vector of indices denoting rows in group one.
rowpen	Glasso penalty for estimating B, the row-row correlation matrix.
B.inv	Optional row-row covariance matrix to be used in GLS in Algorithm 1 prior to model selection centering. If this argument is passed, then it is used instead of estimating the inverse row-row covariance.
rowpen.ModSel	Optional Glasso penalty for estimating B in the second step.
thresh	Threshold for model selection centering. If group means for a column differ by less than the threshold, the column is globally centered rather than group centered. If thresh is NULL, then the theoretically guided threshold is used.

Value

B.hat.inv	Estimated row-row inverse covariance matrix. For identifiability, A and B are scaled so that A has trace m, where m is the number of columns of X.
corr.B.hat.inv	Estimated row-row inverse correlation matrix.
gls.group.means	Matrix with two rows and m columns, where m is the number of columns of X. Entry (i, j) contains the estimated mean of the jth column for an individual in group i, with i = 1,2, and j = 1, ..., m.
gamma.hat	Estimated group mean differences.
test.stats	Vector of test statistics of length m.
p.vals	Vector of two-sided p-values, calculated using the standard normal distribution.
p.vals.adjusted	Vector of p-values, adjusted using the Benjamini-Hochberg fdr adjustment.

Examples

```
# Define sample sizes
n1 <- 5
n2 <- 5
n <- n1 + n2
m <- 200

# Generate data with row and column covariance
# matrices each autorogressive of order 1 with
# parameter 0.2.  The mean is defined so the first
# three columns have true differences in group means
# equal to four.
Z <- matrix(rnorm(m * n), nrow=n, ncol=m)
A <- outer(1:m, 1:m, function(i, j) 0.2^abs(i - j))
B <- outer(1:n, 1:n, function(i, j) 0.2^abs(i - j))
M <- matrix(0, nrow=nrow(Z), ncol=ncol(Z))
group.one.indices <- 1:5
group.two.indices <- 6:10
```

```
M[group.one.indices, 1:3] <- 2
M[group.two.indices, 1:3] <- -2
X <- t(chol(B)) %*% Z %*% chol(A) + M

# Apply Algorithm 2 (jointMeanCovModSelCen) and
# plot the test statistics.
rowpen <- sqrt(log(m) / n)
out <- jointMeanCovModSelCen(X, group.one.indices, rowpen)
plot(out)
summary(out)
```

`jointMeanCovStability` *Estimate Mean and Correlation Structure Using Stability Selection*

Description

Given a data matrix, this function performs stability selection as described in the section "Stability of Gene Sets" in the paper Joint mean and covariance estimation with unreplicated matrix-variate data (2018), M. Hornstein, R. Fan, K. Shedden, and S. Zhou; Journal of the American Statistical Association.

Usage

```
jointMeanCovStability(X, group.one.indices, rowpen,
  n.genes.to.group.center = NULL)
```

Arguments

`X`	Data matrix of size n by m.
`group.one.indices`	Vector of indices denoting rows in group one.
`rowpen`	Glasso penalty for estimating B, the row-row correlation matrix.
`n.genes.to.group.center`	Vector specifying the number of genes to group center on each iteration of the stability selection algorithm. The length of this vector is equal to the number of iterations of stability selection. If this argument is not provided, the default is a decreasing sequence starting with m, followed

Details

Let `m[i]` denote the number of group-centered genes on the ith iteration of stability selection (where `m[i]` is a decreasing sequence). Estimated group means are initialized using unweighted sample means. Then, for each iteration of stability selection: 1. The top `m[i]` genes are selected for group centering by ranking the estimated group differences from the previous iteration. 2. Group means and global means are estimated using GLS, using the inverse row covariance matrix from the previous iteration. The centered data matrix is then used as input to Gemini to estimate the inverse row covariance matrix B.hat.inv. 3. Group means are estimated using GLS with B.hat.inv.

Value

`n.genes.to.group.center`
: Number of group centered genes on each iteration of stability selection.

`betaHat.init`
: Matrix of size 2 by m, containing sample means for each group. Row 1 contains sample means for group one, and row 2 contains sample means for group two.

`gammaHat.init`
: Vector of length m containing differences in sample means.

`B.inv.list`
: List of estimated row-row inverse covariance matrices, where `B.inv.list[[i]]` corresponds to the estimate from the ith iteration of the algorithm, in which the number of group-centered genes is `n.genes.to.group.center[i]`. For identifiability, A and B are scaled so that A has trace m.

`corr.B.inv.list`
: List of inverse correlation matrices corresponding to the inverse covariance matrices `B.inv.list`.

`betaHat`
: List of matrices of size 2 by m, where m is the number of columns of X. For each matrix, entry (i, j) contains the estimated mean of the jth column for an individual in group i, with i = 1,2, and j = 1, ..., m. The matrix `betaHat[[i]]` contains the estimates for the ith iteration of stability selection.

`gamma.hat`
: List of vectors of estimated group mean differences. The vector `gammaHat[[i]]` contains estimates for the ith iteration of stability selection.

`design.effecs`
: Vector containing the estimated design effect for each iteration of stability selection.

`gls.test.stats`
: List of vectors of test statistics for each iteration of stability selection.

`p.vals`
: List of vectors of two-sided p-values, calculated using the standard normal distribution.

`p.vals.adjusted`
: List of vectors of p-values, adjusted using the Benjamini-Hochberg fdr adjustment.

Examples

```
# Generate matrix-variate data.
n1 <- 5
n2 <- 5
n <- n1 + n2
group.one.indices <- 1:5
group.two.indices <- 6:10
m <- 20
M <- matrix(0, nrow=n, ncol=m)
# In this example, the first three variables have nonzero
# mean differences.
M[1:n1, 1:3] <- 3
M[(n1 + 1):n2, 1:3] <- -3
X <- matrix(rnorm(n * m), nrow=n, ncol=m) + M

# Apply the stability algorithm.
rowpen <- sqrt(log(m) / n)
n.genes.to.group.center <- c(10, 5, 2)
```

```
out <- jointMeanCovStability(
 X, group.one.indices, rowpen, c(2e3, n.genes.to.group.center))

# Make quantile plots of the test statistics for each
# iteration of the stability algorithm.
opar <- par(mfrow=c(2, 2), pty="s")
qqnorm(out$gammaHat.init,
  main=sprintf("%d genes group centered", m))
abline(a=0, b=1)
qqnorm(out$gammaHat[[1]],
  main=sprintf("%d genes group centered",
   n.genes.to.group.center[1]))
abline(a=0, b=1)
qqnorm(out$gammaHat[[2]],
  main=sprintf("%d genes group centered",
   n.genes.to.group.center[2]))
abline(a=0, b=1)
qqnorm(out$gammaHat[[3]],
  main=sprintf("%d genes group centered",
   n.genes.to.group.center[3]))
abline(a=0, b=1)
par(opar)
```

`plot.jointMeanCov` *Quantile Plot of Test Statistics*

Description

This function displays a quantile plot of test statistics, based on the output of the functions `jointMeanCovGroupCen` or `jointMeanCovModSelCen`.

Usage

```
## S3 method for class 'jointMeanCov'
plot(x, ...)
```

Arguments

`x`	output of `jointMeanCovGroupCen` or `jointMeanCovModSelCen`.
`...`	other plotting arguments passed to `qqnorm`.

Examples

```
# Define sample sizes
n1 <- 5
n2 <- 5
n <- n1 + n2
m <- 200
```

```
# Generate data with row and column covariance
# matrices each autorogressive of order 1 with
# parameter 0.2.  The mean is defined so the first
# three columns have true differences in group means
# equal to four.
Z <- matrix(rnorm(m * n), nrow=n, ncol=m)
A <- outer(1:m, 1:m, function(i, j) 0.2^abs(i - j))
B <- outer(1:n, 1:n, function(i, j) 0.2^abs(i - j))
M <- matrix(0, nrow=nrow(Z), ncol=ncol(Z))
group.one.indices <- 1:5
group.two.indices <- 6:10
M[group.one.indices, 1:3] <- 2
M[group.two.indices, 1:3] <- -2
X <- t(chol(B)) %*% Z %*% chol(A) + M

# Apply Algorithm 2 (jointMeanCovModSelCen) and plot the
# test statistics.
rowpen <- sqrt(log(m) / n)
out <- jointMeanCovModSelCen(X, group.one.indices, rowpen)
plot(out)
```

---

summary.jointMeanCov *Summary of Test Statistics*

Description

summary method for class jointMeanCov. This function displays the minimum, maximum, mean, median, 25th percentile, and 75th percentile of the test statistics.

Usage

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

Arguments

object	output of jointMeanCovGroupCen or jointMeanCovModSelCen.
...	other arguments passed to summary.

Examples

```
# Define sample sizes
n1 <- 5
n2 <- 5
n <- n1 + n2
m <- 200

# Generate data with row and column covariance
# matrices each autorogressive of order 1 with
```

```
# parameter 0.2.  The mean is defined so the first
# three columns have true differences in group means
# equal to four.
Z <- matrix(rnorm(m * n), nrow=n, ncol=m)
A <- outer(1:m, 1:m, function(i, j) 0.2^abs(i - j))
B <- outer(1:n, 1:n, function(i, j) 0.2^abs(i - j))
M <- matrix(0, nrow=nrow(Z), ncol=ncol(Z))
group.one.indices <- 1:5
group.two.indices <- 6:10
M[group.one.indices, 1:3] <- 2
M[group.two.indices, 1:3] <- -2
X <- t(chol(B)) %*% Z %*% chol(A) + M

# Apply Algorithm 2 (jointMeanCovModSelCen) and pass the
# output to the summary function.
rowpen <- sqrt(log(m) / n)
out <- jointMeanCovModSelCen(X, group.one.indices, rowpen)
summary(out)
```

`theoryRowpenUpperBound` *Penalty Parameter for Covariance Estimation Based on Theory*

Description

This function returns a theoretically-guided choice of the glasso penalty parameter, based on both the row and column covariance matrices.

Usage

```
theoryRowpenUpperBound(A, B, n1, n2)
```

Arguments

`A`	column covariance matrix.
`B`	row covariance matrix.
`n1`	sample size of group one.
`n2`	sample size of group two.

Value

Returns a theoretically guided choice of the glasso penalty parameter.

References

Joint mean and covariance estimation with unreplicated matrix-variate data Michael Hornstein, Roger Fan, Kerby Shedden, Shuheng Zhou (2018). Joint mean and covariance estimation with unreplicated matrix-variate data. Journal of the American Statistical Association

Examples

```
# Define sample sizes
n1 <- 10
n2 <- 10
n <- n1 + n2
m <- 2e3
# Column covariance matrix (autoregressive of order 1)
A <- outer(1:n, 1:n, function(x, y) 0.2^abs(x - y))
# Row covariance matrix (autoregressive of order 1)
B <- outer(1:n, 1:n, function(x, y) 0.8^abs(x - y))
# Calculate theoretically guided Gemini penalty.
rowpen <- theoryRowpenUpperBound(A, B, n1, n2)
print(rowpen)
```

---

theoryRowpenUpperBoundDiagA

Penalty Parameter for Covariance Estimation Based on Theory

Description

This function returns a theoretically-guided choice of the glasso penalty parameter, treating the column correlation matrix as the identity.

Usage

```
theoryRowpenUpperBoundDiagA(B, n1, n2, m)
```

Arguments

B	row covariance matrix.
n1	sample size of group one.
n2	sample size of group two.
m	number of columns of the data matrix (where the data matrix is of size n by m, with n = n1 + n2).

Value

Returns a theoretically guided choice of the glasso penalty parameter.

References

Joint mean and covariance estimation with unreplicated matrix-variate data Michael Hornstein, Roger Fan, Kerby Shedden, Shuheng Zhou (2018). Joint mean and covariance estimation with unreplicated matrix-variate data. Journal of the American Statistical Association

Examples

```
# Define sample sizes
n1 <- 10
n2 <- 10
n <- n1 + n2
m <- 2e3
# Row covariance matrix (autoregressive of order 1)
B <- outer(1:n, 1:n, function(x, y) 0.8^abs(x - y))
# Calculate theoretically guided Gemini penalty.
rowpen <- theoryRowpenUpperBoundDiagA(B, n1, n2, m)
print(rowpen)
```

---

twoGroupDesignMatrix *Design Matrix for Two-Group Mean Estimation*

Description

This function returns the design matrix for two-group mean estimation. The first column contains indicators for membership in the first group, and the second column contains indicators for memebership in the second group.

Usage

```
twoGroupDesignMatrix(group.one.indices, group.two.indices)
```

Arguments

group.one.indices	indices of observations in group one.
group.two.indices	indices of observations in group two.

Details

Example

```
D <- twoGroupDesignMatrix(1:2, 3:5)
# print(D) displays the following:
     [,1] [,2]
[1,]    1    0
[2,]    1    0
[3,]    0    1
[4,]    0    1
[5,]    0    1
```

Value

Returns a design matrix of size n by 2, where n is the sample size.

Index