Package ‘BlockMissingData’

September 12, 2023

Type Package

Title Integrating Multi-Source Block-Wise Missing Data in Model Selection

Version 0.1.0

Author Fei Xue [aut, cre],
Annie Qu [aut]

Maintainer Fei Xue <feixue@purdue.edu>


License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.2.3

Imports doParallel, foreach, glmnet, glmnetcr, MASS, Matrix, pryr, stats

NeedsCompilation no

Repository CRAN

Date/Publication 2023-09-12 08:22:35 UTC

R topics documented:

imputeglm.predict .................................................. 2
MBI ........................................................................ 4

Index 8
impute glm.predict

**Imputation using generalized linear models for missing values**

**Description**

The function performs imputation using generalized linear models for missing values in a dataset. It fits these models for each specified response variable separately, utilizing other specified variables, and returns the estimated coefficients and predicted values for each variable. The function handles different distribution families, such as Gaussian, Binomial, and Ordinal, for GLM estimation.

**Usage**

```r
impute glm.predict(X, ind_y, ind_x = -ind_y, miss, newdata, family = "gaussian")
```

**Arguments**

- `X`: Data matrix containing all the variables that may contain missing values.
- `ind_y`: A vector specifying the indices of response variables in the dataset.
- `ind_x`: A vector specifying the indices of predictor variables in the dataset. By default, it is set to `-ind_y`, which means all variables other than the response variables are considered as predictors.
- `miss`: A logical matrix indicating the missing values in the dataset.
- `newdata`: Data matrix for which imputed values are required. It should have the same column names as the original dataset.
- `family`: A character indicating the distribution family of the GLM. Possible values are "gaussian" (default), "binomial", and "ordinal".

**Value**

- `B`: A matrix of estimated coefficients, where each column contains the coefficients for a response variable, and each row corresponds to a predictor variable (including the intercept term).
- `PRED`: A matrix of predicted values (or imputations), where each column contains the predicted values for a response variable, and each row corresponds to an observation in the newdata (if provided).

**Author(s)**

Fei Xue and Annie Qu
Examples

library(MASS)

# Number of subjects
n <- 700

# Number of total covariates
p <- 40

# Number of missing groups of subjects
ngroup <- 4

# Number of data sources
nsource <- 4

# Starting indexes of covariates in data sources
cov_index=c(1, 13, 25, 37)

# Starting indexes of subjects in missing groups
sub_index=c(1, 31, 251, 471)

# Indexes of missing data sources in missing groups, respectively ('NULL' represents no missing)
miss_source=list(NULL, 3, 2, 1)

# Create a design matrix
set.seed(1)
sigma=diag(1-0.4,p,p)+matrix(0.4,p,p)
X <- mvrnorm(n,rep(0,p),sigma)

# Introduce some block-wise missing
for (i in 1:ngroup) {
  if (!is.null(miss_source[[i]])) {
    if (i==ngroup) {
      if (miss_source[[i]]==nsource) {
        X[sub_index[[i]]:n, cov_index[miss_source[[i]]]:p] = NA
      } else {
        X[sub_index[[i]]:n, cov_index[miss_source[[i]]]:(cov_index[miss_source[[i]]]+1)-1] = NA
      }
    } else {
      if (miss_source[[i]]==nsource) {
        X[sub_index[[i]]:(sub_index[[i+1]]-1), cov_index[miss_source[[i]]]:p] = NA
      } else {
        X[sub_index[[i]]:(sub_index[[i+1]]-1), cov_index[miss_source[[i]]]:
          (cov_index[miss_source[[i]]]+1)-1] = NA
      }
    }
  }
}

# Define missing data pattern
miss <- is.na(X)
# Choose response and predictor variables
ind_y <- 25:36
ind_x <- 13:24

# Data that need imputation
newdata <- X[31:250,]

# Use the function
result <- imputeglm.predict(X = X, ind_y = ind_y, ind_x = ind_x, miss = miss, newdata = newdata)

---

### MBI

**Variable selection method with multiple block-wise imputation (MBI)**

#### Description

Fit a variable selection method with multiple block-wise imputation (MBI).

#### Usage

```r
MBI(
  X,
  y,
  cov_index,
  sub_index,
  miss_source,
  complete,
  lambda = NULL,
  eps1 = 0.001,
  eps2 = 1e-07,
  eps3 = 1e-08,
  max.iter = 1000,
  lambda.min = ifelse(n > p, 0.001, 0.05),
  nlam = 100,
  beta0 = NULL,
  a = 3.7,
  gamma.ebic = 0.5,
  alpha1 = 0.5^(0:12),
  h1 = 2^(-(8:30)),
  ratio = 1
)
```

#### Arguments

- **X**: Design matrix for block-wise missing covariates.
- **y**: Response vector.
- **cov_index**: Starting indexes of covariates in data sources.
- **sub_index**: Starting indexes of subjects in missing groups.
- **miss_source**: Indexes of missing data sources in missing groups, respectively (‘NULL’ represents no missing).
complete Logical indicator of whether there is a group of complete cases. If there is a group of complete cases, it should be the first group. ‘TRUE’ represents that there is a group of complete cases.

lambda A user supplied sequence of tuning parameter in penalty. If NULL, a sequence is automatically generated.

eps1 Convergence threshold at certain stage of the algorithm. Default is 1e-3.

eps2 Convergence threshold at certain stage of the algorithm. Default is 1e-7.

eps3 Convergence threshold at certain stage of the algorithm. Default is 1e-8.

max.iter The maximum number of iterations allowed. Default is 1000.

lambda.min Smallest value for lambda, as a fraction of the maximum value in lambda. Default depends on the size of input.

nlam The number of lambda values. Default is 100.

beta0 Initial value for regression coefficients. If NULL, they are initialized automatically.

a Tuning parameter in the SCAD penalty. Default is 3.7.

gamma.ebic Parameter in the EBIC criterion. Default is 0.5.

alpha1 A sequence of candidate values for the step size in the conjugate gradient algorithm. Default is 0.5^(0:12).

h1 A sequence of candidate values for the parameter in the numerical calculation of the first derivative of the objective function. Default is 2^(-8:30)).

ratio Parameter in the numerical calculation of the first derivative. Default is 1.

Details
The function uses the penalized generalized method of moments with multiple block-wise imputation to handle block-wise missing data, commonly found in multi-source datasets.

Value

beta Estimated coefficients matrix with length(lambda) rows and dim(X)[2] columns.

lambda The actual sequence of lambda values used.

bic1 BIC criterion values. ‘0’ should be ignored.

notcon Value indicating whether the algorithm is converged or not. ‘0’ represents convergence; otherwise non-convergence.

intercept Intercept sequence of length length(lambda).

beta0 Estimated coefficients matrix for standardized X

Author(s)
Fei Xue and Annie Qu

References
Examples

library(MASS)

# Number of subjects
n <- 30

# Number of total covariates
p <- 4

# Number of missing groups of subjects
ngroup <- 2

# Number of data sources
nsource <- 2

# Starting indexes of covariates in data sources
cov_index=c(1, 3)

# Starting indexes of subjects in missing groups
sub_index=c(1, 16)

# Indexes of missing data sources in missing groups, respectively ('NULL' represents no missing)
mis_source=list(NULL, 1)

# Indicator of whether there is a group of complete cases. If there is a group of complete cases,
# it should be the first group.
complete=TRUE

# Create a block-wise missing design matrix X and response vector y
set.seed(1)
sigma=diag(1-0.4,p,p)+matrix(0.4,p,p)
X <- mvrnorm(n,rep(0,p),sigma)
beta_true <- c(2.5, 0, 3, 0)
y <- rnorm(n) + X%*%beta_true

for (i in 1:ngroup) {
  if (!is.null(mis_source[[i]])) {
    if (i==ngroup) {
      if (mis_source[[i]]==nsource) {
        X[sub_index[i]:n, cov_index[mis_source[[i]]]:p] = NA
      } else {
        X[sub_index[i]:n, cov_index[mis_source[[i]]]: (cov_index[mis_source[[i]]]+1)-1] = NA
      }
    } else {
      if (mis_source[[i]]==nsource) {
        X[sub_index[i]: (sub_index[i+1]-1), cov_index[mis_source[[i]]]:p] = NA
      } else {
        X[sub_index[i]: (sub_index[i+1]-1), cov_index[mis_source[[i]]]: (cov_index[mis_source[[i]]]+1)-1] = NA
      }
    }
  }
}
# Now we can use the function with this simulated data

```r
#start.time = proc.time()
result <- MBI(X=X, y=y, cov_index=cov_index, sub_index=sub_index, miss_source=miss_source,
complete=complete, nlam = 15, eps2 = 1e-3, h1=2^(-(8:20)))
#time = proc.time() - start.time

theta=result$beta
bic1=result$bic1
best=which.min(bic1[bic1!=0])
beta_est=theta[best,]
```
Index

impute glm.predict, 2

MBI, 4