# Package 'GENMETA'

October 17, 2018

**Title** Implements Generalized Meta-Analysis

**Version** 0.1

**Description** Generalized meta-analysis using
iterated-reweighted least square algorithm.

**Depends** R (>= 2.15.3), MASS, graphics

**Imports** Matrix, magic

**License** GPL

**Encoding** UTF-8

**LazyData** true

**RoxygenNote** 6.1.0

**Collate** 'myoptim.R' 'sign.star.R' 'GENMETA.R' 'GENMETA.control.R'
'GENMETA.summary.R' 'GENMETA.plot.R'

**NeedsCompilation** no

**Author** Prosenjit Kundu [aut, cre],
Runlong Tang [aut],
Nilanjan Chatterjee [aut]

**Maintainer** Prosenjit Kundu <pkundu@jhu.edu>

## R topics documented:



---

GENMETA *Implementing Generalized Meta-analysis*

---

### Description

Generalized Meta-analysis(GENMETA) is an approach for combining information on multivariate regression parameters across multiple different studies which have different, but, possibly overlapping information on subsets of covariates. GENMETA implements the generalized meta-analysis using IRWLS algorithm.

## Usage

```
GENMETA(study_info, ref_dat, model, variable_intercepts = FALSE,
  initial_val = NULL, control = list(epsilon = 1e-06, maxit = 1000))
```

## Arguments

| | |
|---|---|
| study_info | a list of lists containing information about the studies; the main list contains a list for each study, which must have the fields:<br>• "Coeff": a named numeric vector containing the estimates of regression parameters (including intercept) where the names identify the covariates. For example, names(study_info$Coeff) <- c("(Intercept)", "Age", "Height", "Weight").<br>• "Covariance": a matrix containing an estimate of variance-covariance matrix of the regression parameters. This can be NULL if the "Sample_size" is provided.<br>• "Sample_size": a numeric containing sample size of the study. This can be NULL if the "Covariance" is provided. |
| ref_dat | a data matrix containing all the distinct covariates across studies from a reference set of individuals. This is used for estimating joint distribution of the covariates. The data matrix must have the vector of ones as its first column. The column names of the data matrix should match the names of the covariates from the studies. |
| model | a description of the type of regression model; this is a character string naming the regression model. The current version is for "logistic" and "linear". |
| variable_intercepts | an optional logical (applicable only when the model is "logistic"); if TRUE, the intercepts of the true models for each of the studies are assumed to be different. Default is FALSE. |
| initial_val | an optional numeric vector containing initial values for the maximal model parameters which is needed for the IRWLS algorithm. Default is set to the one obtained from standard meta-analysis of the available estimates for each of the parameters across studies. |
| control | an optional list containing the epsilon (positive numeric) and maxiter (positive number) needed for convergence of the algorithm. Default epsilon and maximum iterations are 1e-06 and 1000, respectively. For creating a control argument for GENMETA, see GENMETA.control. |

## Details

Generalized Meta-analysis (GENMETA) is a tool that allows researchers to quickly build models for multivariate meta-analysis in the presence of disparate covariate information across studies. It is implemented based on mainly two input arguments:

- Information on the model parameters from each of the studies.
- Reference data for estimation of the joint distribution of all the distinct covariates across studies.

The software provides flexibility to the users to choose the intercepts to be different (when the model is logistic) across studies through the input argument, variable_intercepts. It also allows estimation of the regression parameters, only from the sample sizes of the studies when it is difficult to obtain estimate of the variance-covariance matrices.

**Note**: GENMETA will not work if both the estimates of the covariance matrix and the sample size are NULL.

When the model is "linear", it is assumed that the outcome is standardized to have unit variance. For more details on the IRWLS, see References.

## Value

An object of class "GENMETA" is a list containing GENMETA estimate, its variance-covariance matrix and estimates the residual variance in the case of "linear" model .

| | |
|---|---|
| `Est.coeff` | a numeric vector containing the estimated regression coefficients of the maximal model using optimal weighting matrix. |
| `Est.var.cov` | a matrix containing estimate of variance-covariance matrix of the corresponding GENMETA estimator. |
| `Res.var` | a numeric containing the residual variance of the maximal model when it is linear. It is calculated from the formula : $1 - \hat{\beta}^T_{GENMETA} var(X) \hat{\beta}_{GENMETA}$ which is derived by assuming the outcomes to have unit variance. $var(X)$ is calculated from reference data. Res.var is NA when the model is "logistic". |
| `iter` | a numeric containing the number of iterations used in the algorithm |
| `call` | the matched call |

The function `GENMETA.summary` prints a summary of the results obtained from GENMETA.
The function `GENMETA.plot` plots the estimate of the parameter from each of the studies, the summary measure(GENMETA estimate) and their confidence intervals.

## Author(s)

Prosenjit Kundu, Runlong Tang and Nilanjan Chatterjee.

## References

Tang, R., Kundu, P. and Chatterjee, N. (2017) Generalized Meta-Analysis for Multivariate Regression Models Across Studies with Disparate Covariate Information. arXiv:1708.03818v1 [stat.ME].

## See Also

`GENMETA.summary`, `GENMETA.plot`.

## Examples

```
# This example shows how to create the inputs GENMETA and then implement generalized meta-analysis
#########################
### Basic setting #######
#########################
d.X = 3 # number of covariates.
mu = matrix(rep(0,d.X), nrow=d.X) # mean vector of the covariates.
r1 = 0.3 # correlation coefficient of the covariates.
r2 = 0.6
r3 = 0.1
Sigma = matrix(
  c(1, r1, r2,
    r1, 1, r3,
```

```
    r2, r3, 1),
  nrow=d.X,
  ncol=d.X) # covariance matrix of the covariates.
beta.star = matrix(c(-1.2, log(1.3), log(1.3), log(1.3)),nrow = d.X+1) # beta.star
#beta.star = matrix(c(-1.2, 0.26, 0.26, 0.26),nrow = d.X+1) # beta.star
#beta.star = matrix(c(-3, 1, 2, 3),nrow = d.X+1) # beta.star
n1 = 300 # sample size of the 1st data set.
n2 = 500 # 2nd
n3 = 1000 # 3rd
n = 50
sim=1
set.seed(sim)
X.rf = MASS::mvrnorm(n = n, mu, Sigma)
X.m1 = MASS::mvrnorm(n = n1, mu, Sigma) # Generate the covariates.
X.m1.1 = cbind(rep(1, n1), X.m1) # Add a column of 1's to X.m1.
p.m1 = 1/(1+exp(-X.m1.1%*%beta.star)) # the vector of probabilities
Y.m1 = rbinom(n1, size=1, p.m1) # the Bernoulli responses
# print(p.m1[1])
# print(mean(Y.m1))
# print(mean(p.m1))
# Generate data set 2. m1 means model 2.
X.m2 = MASS::mvrnorm(n = n2, mu, Sigma)
X.m2.1 = cbind(rep(1, n2), X.m2)
p.m2 = 1/(1+exp(-X.m2.1%*%beta.star))
Y.m2 = rbinom(n2, size=1, p.m2)
X.m3 = MASS::mvrnorm(n = n3, mu, Sigma)
X.m3.1 = cbind(rep(1, n3), X.m3)
p.m3 = 1/(1+exp(-X.m3.1%*%beta.star))
Y.m3 = rbinom(n3, size=1, p.m3)
#####
### Create data sets in the format of data frame.
#####
data.m1 = data.frame(Y=Y.m1, X.m1)
data.m2 = data.frame(Y=Y.m2, X.m2)
data.m3 = data.frame(Y=Y.m3, X.m3)
#####
### Apply logistic regression with reduced models to the data sets
#####
logit.m1 <- glm(Y ~ X1 + X2, data = data.m1, family = "binomial")
# print(logit.m1)
if(logit.m1$converged == FALSE)
{
  print("glm for logit.m1 is not convergent.")
  next
}
logit.m2 <- glm(Y ~ X2 + X3, data = data.m2, family = "binomial")
# print(logit.m2)
if(logit.m2$converged == FALSE)
{
  print("glm for logit.m2 is not convergent.")
  next
}
logit.m3 <- glm(Y ~ X1 + X3, data = data.m3, family = "binomial")
# print(logit.m3)
if(logit.m3$converged == FALSE)
{
  print("glm for logit.m3 is not convergent.")
```

```
  next
}
#####
### Obtain the estimators of the parameters in the reduced models.
#####
theta.m1 = logit.m1$coefficients
theta.m2 = logit.m2$coefficients
theta.m3 = logit.m3$coefficients
#####
### Find the covariance matrix estimators for the reduced models
#####
#####
# Basic notations for inputs
#####
K = 3 # Number of data sets
A1 = c(1, 2) # index set A1, the indexes of the covariates of data set 1.
A2 = c(2, 3) # index set A2
A3 = c(1, 3) # index set A3
X.m1.used = cbind(rep(1, n1), X.m1[, A1, drop=FALSE])
X.m2.used = cbind(rep(1, n2), X.m2[, A2, drop=FALSE])
X.m3.used = cbind(rep(1, n3), X.m3[, A3, drop=FALSE])
# str(X.m1.used)
# str(X.m2.used)
# str(X.m3.used)
##### Find Sigma.m1
T.1 = matrix(rep(0, (length(A1)+1)^2), nrow=length(A1)+1)
T.2 = T.1
for (i in 1:n1)
{
a = as.vector(exp(-X.m1.used[i, , drop=FALSE]%*%theta.m1))
  T.1 = T.1 + (a/(1+a)^2) * (t(X.m1.used[i, , drop=FALSE])%*%X.m1.used[i, , drop=FALSE])
}
for (i in 1:n1)
{
  a = as.vector(1/( 1 + exp(-X.m1.used[i, , drop=FALSE]%*%theta.m1)))
  T.2 = T.2 + (Y.m1[i]-a)^2 * (t(X.m1.used[i, , drop=FALSE])%*%X.m1.used[i, , drop=FALSE])
}
Sigma.m1 = solve(T.1)%*%T.2%*%solve(T.1) # This is actually Sigma.m1.n1.
##### Find Sigma.m2
T.1 = matrix(rep(0, (length(A2)+1)^2), nrow=length(A2)+1)
T.2 = T.1
for (i in 1:n2)
{
  a = as.vector(exp(-X.m2.used[i, , drop=FALSE]%*%theta.m2))
  T.1 = T.1 + (a/(1+a)^2) * (t(X.m2.used[i, , drop=FALSE])%*%X.m2.used[i, , drop=FALSE])
}
for (i in 1:n2)
{
  a = as.vector(1/( 1 + exp(-X.m2.used[i, , drop=FALSE]%*%theta.m2)))
  T.2 = T.2 + (Y.m2[i]-a)^2 * (t(X.m2.used[i, , drop=FALSE])%*%X.m2.used[i, , drop=FALSE])
}
Sigma.m2 = solve(T.1)%*%T.2%*%solve(T.1)
##### Find Sigma.m3
T.1 = matrix(rep(0, (length(A3)+1)^2), nrow=length(A3)+1)
T.2 = T.1
for (i in 1:n3)
{
```

```
  a = as.vector(exp(-X.m3.used[i, , drop=FALSE]%*%theta.m3))
  T.1 = T.1 + (a/(1+a)^2) * (t(X.m3.used[i, , drop=FALSE])%*%X.m3.used[i, , drop=FALSE])
}
for (i in 1:n3)
{
  a = as.vector(1/( 1 + exp(-X.m3.used[i, , drop=FALSE]%*%theta.m3)))
  T.2 = T.2 + (Y.m3[i]-a)^2 * (t(X.m3.used[i, , drop=FALSE])%*%X.m3.used[i, , drop=FALSE])
}
Sigma.m3 = solve(T.1)%*%T.2%*%solve(T.1)
names(theta.m1)=c("(Intercept)","Age","Height")
names(theta.m2)=c("(Intercept)","Height", "Weight")
names(theta.m3)=c("(Intercept)","Age", "Weight")
study1 = list(Coeff=theta.m1,Covariance=NULL,Sample_size=n1)
study2 = list(Coeff=theta.m2,Covariance=NULL,Sample_size=n2)
study3 = list(Coeff=theta.m3,Covariance=NULL,Sample_size=n3)
studies = list(study1,study2,study3)
model = "logistic"
reference = cbind(rep(1,n), X.rf)
colnames(reference) = c("(Intercept)","Age","Height", "Weight")
same.inter = GENMETA(studies, reference, model, initial_val = c(-1.2, log(1.3), log(1.3), log(1.3)))
diff.inter = GENMETA(studies, reference, model, variable_intercepts=TRUE)
```

---

`GENMETA.control` *Auxiliary for controlling the IRWLS algorithm*

---

## Description

This is an auxiliary function for the iteratively reweighted least squares algorithm for GMeta. This is used internally by the myoptim function, but can be used by the user to create a control argument in the GENMETA function

## Usage

```
GENMETA.control(epsilon = 1e-06, maxit = 1000)
```

## Arguments

| | |
|---|---|
| `epsilon` | a positive numeric indicating converegence tolerence; the algorithm stops when the absolute difference between the estimates in current and previous step is less than epsilon, i.e, $\|estimate_{new} - estimate_{old}\| < \epsilon$ |
| `maxit` | a positive number indicating the maximum number of iterations to be used in the algorithm. Default is 1000. |

## Value

A list with components named as the arguments.

## Examples

```
control <- GENMETA.control(1e-08, 100)
```

GENMETA.plot *Generalized Meta-analysis(forest plot)*

## Description

This function plots the confidence intervals with boxes as the study specific estimates and diamond as the GMeta estimate. For the current version, it assumes that the estimate of the variance-covariance matrix in each of the studies is provided. It is demonstrated using a different dataset, "study_info_plot", which meets the assumption.

## Usage

```
GENMETA.plot(x, study_info_plot)
```

## Arguments

| | |
|---|---|
| x | an object of class "GENMETA" |
| study_info_plot | a list of lists containing information about the studies(similar to the study_info argument used in GENMETA function.) |

## Examples

```
# This example shows how to obtain the forest plot of GENMETA object.
#########################
### Basic setting #######
#########################
d.X = 3 # number of covariates.
mu = matrix(rep(0,d.X), nrow=d.X) # mean vector of the covariates.
r1 = 0.3 # correlation coefficient of the covariates.
r2 = 0.6
r3 = 0.1
Sigma = matrix(
  c(1, r1, r2,
    r1, 1, r3,
    r2, r3, 1),
  nrow=d.X,
  ncol=d.X) # covariance matrix of the covariates.
beta.star = matrix(c(-1.2, log(1.3), log(1.3), log(1.3)),nrow = d.X+1)
#beta.star = matrix(c(-1.2, 0.26, 0.26, 0.26),nrow = d.X+1) # beta.star
#beta.star = matrix(c(-3, 1, 2, 3),nrow = d.X+1) # beta.star
n1 = 300 # sample size of the 1st data set.
n2 = 500 # 2nd
n3 = 1000 # 3rd
n = 50
sim=1
set.seed(sim)
X.rf = MASS::mvrnorm(n = n, mu, Sigma)
X.m1 = MASS::mvrnorm(n = n1, mu, Sigma) # Generate the covariates.
X.m1.1 = cbind(rep(1, n1), X.m1) # Add a column of 1's to X.m1.
p.m1 = 1/(1+exp(-X.m1.1%*%beta.star)) # the vector of probabilities
Y.m1 = rbinom(n1, size=1, p.m1) # the Bernoulli responses
# print(p.m1[1])
```

```
# print(mean(Y.m1))
# print(mean(p.m1))
# Generate data set 2. m1 means model 2.
X.m2 = MASS::mvrnorm(n = n2, mu, Sigma)
X.m2.1 = cbind(rep(1, n2), X.m2)
p.m2 = 1/(1+exp(-X.m2.1%*%beta.star))
Y.m2 = rbinom(n2, size=1, p.m2)
X.m3 = MASS::mvrnorm(n = n3, mu, Sigma)
X.m3.1 = cbind(rep(1, n3), X.m3)
p.m3 = 1/(1+exp(-X.m3.1%*%beta.star))
Y.m3 = rbinom(n3, size=1, p.m3)
#####
### Create data sets in the format of data frame.
#####
data.m1 = data.frame(Y=Y.m1, X.m1)
data.m2 = data.frame(Y=Y.m2, X.m2)
data.m3 = data.frame(Y=Y.m3, X.m3)
#####
### Apply logistic regression with reduced models to the data sets
#####
logit.m1 <- glm(Y ~ X1 + X2, data = data.m1, family = "binomial")
# print(logit.m1)
if(logit.m1$converged == FALSE)
{
  print("glm for logit.m1 is not convergent.")
  next
}
logit.m2 <- glm(Y ~ X2 + X3, data = data.m2, family = "binomial")
# print(logit.m2)
if(logit.m2$converged == FALSE)
{
  print("glm for logit.m2 is not convergent.")
  next
}
logit.m3 <- glm(Y ~ X1 + X3, data = data.m3, family = "binomial")
# print(logit.m3)
if(logit.m3$converged == FALSE)
{
  print("glm for logit.m3 is not convergent.")
  next
}
#####
### Obtain the estimators of the parameters in the reduced models.
#####
theta.m1 = logit.m1$coefficients
theta.m2 = logit.m2$coefficients
theta.m3 = logit.m3$coefficients
#####
### Find the covariance matrix estimators for the reduced models
#####
#####
# Basic notations for inputs
#####
K = 3 # Number of data sets
A1 = c(1, 2) # index set A1, the indexes of the covariates of data set 1.
A2 = c(2, 3) # index set A2
A3 = c(1, 3) # index set A3
```

```
X.m1.used = cbind(rep(1, n1), X.m1[, A1, drop=FALSE])
X.m2.used = cbind(rep(1, n2), X.m2[, A2, drop=FALSE])
X.m3.used = cbind(rep(1, n3), X.m3[, A3, drop=FALSE])
# str(X.m1.used)
# str(X.m2.used)
# str(X.m3.used)
##### Find Sigma.m1
T.1 = matrix(rep(0, (length(A1)+1)^2), nrow=length(A1)+1)
T.2 = T.1
for (i in 1:n1)
{
a = as.vector(exp(-X.m1.used[i, , drop=FALSE]%*%theta.m1))
  T.1 = T.1 + (a/(1+a)^2) * (t(X.m1.used[i, , drop=FALSE])%*%X.m1.used[i, , drop=FALSE])
}
for (i in 1:n1)
{
  a = as.vector(1/( 1 + exp(-X.m1.used[i, , drop=FALSE]%*%theta.m1)))
  T.2 = T.2 + (Y.m1[i]-a)^2 * (t(X.m1.used[i, , drop=FALSE])%*%X.m1.used[i, , drop=FALSE])
}
Sigma.m1 = solve(T.1)%*%T.2%*%solve(T.1) # This is actually Sigma.m1.n1.
##### Find Sigma.m2
T.1 = matrix(rep(0, (length(A2)+1)^2), nrow=length(A2)+1)
T.2 = T.1
for (i in 1:n2)
{
  a = as.vector(exp(-X.m2.used[i, , drop=FALSE]%*%theta.m2))
  T.1 = T.1 + (a/(1+a)^2) * (t(X.m2.used[i, , drop=FALSE])%*%X.m2.used[i, , drop=FALSE])
}
for (i in 1:n2)
{
  a = as.vector(1/( 1 + exp(-X.m2.used[i, , drop=FALSE]%*%theta.m2)))
  T.2 = T.2 + (Y.m2[i]-a)^2 * (t(X.m2.used[i, , drop=FALSE])%*%X.m2.used[i, , drop=FALSE])
}
Sigma.m2 = solve(T.1)%*%T.2%*%solve(T.1)
##### Find Sigma.m3
T.1 = matrix(rep(0, (length(A3)+1)^2), nrow=length(A3)+1)
T.2 = T.1
for (i in 1:n3)
{
  a = as.vector(exp(-X.m3.used[i, , drop=FALSE]%*%theta.m3))
  T.1 = T.1 + (a/(1+a)^2) * (t(X.m3.used[i, , drop=FALSE])%*%X.m3.used[i, , drop=FALSE])
}
for (i in 1:n3)
{
  a = as.vector(1/( 1 + exp(-X.m3.used[i, , drop=FALSE]%*%theta.m3)))
  T.2 = T.2 + (Y.m3[i]-a)^2 * (t(X.m3.used[i, , drop=FALSE])%*%X.m3.used[i, , drop=FALSE])
}
Sigma.m3 = solve(T.1)%*%T.2%*%solve(T.1)
names(theta.m1)=c("(Intercept)","Age","Height")
names(theta.m2)=c("(Intercept)","Height", "Weight")
names(theta.m3)=c("(Intercept)","Age", "Weight")
study1 = list(Coeff=theta.m1,Covariance=Sigma.m1,Sample_size=n1)
study2 = list(Coeff=theta.m2,Covariance=Sigma.m2,Sample_size=n2)
study3 = list(Coeff=theta.m3,Covariance=Sigma.m3,Sample_size=n3)
studies = list(study1,study2,study3)
model = "logistic"
reference = cbind(rep(1,n), X.rf)
```

```
colnames(reference) = c("(Intercept)","Age","Height", "Weight")
result_diff <- GENMETA(studies, reference, model, variable_intercepts = TRUE)
GENMETA.plot(result_diff, studies)
```

---

GENMETA.summary *Summarizing Generalized Meta Analysis*

---

## Description

This function prints the summary of GENMETA results.

## Usage

```
GENMETA.summary(object, signi_digits = 3)
```

## Arguments

| | |
|---|---|
| object | an object of class "GENMETA" |
| signi_digits | an optional numeric indicating the number of significant digits to be shown in the summary. Default is 3. |

## Examples

```
# This example shows how to obtain the summary of GENMETA object.
#####
### Basic setting
#####
d.X = 3 # number of covariates.
mu = matrix(rep(0,d.X), nrow=d.X) # mean vector of the covariates.
r1 = 0.3 # correlation coefficient of the covariates.
r2 = 0.6
r3 = 0.1
Sigma = matrix(
  c(1, r1, r2,
    r1, 1, r3,
    r2, r3, 1),
  nrow=d.X,
  ncol=d.X) # covariance matrix of the covariates.
beta.star = matrix(c(-1.2, log(1.3), log(1.3), log(1.3)),nrow = d.X+1) # beta.star
#beta.star = matrix(c(-1.2, 0.26, 0.26, 0.26),nrow = d.X+1) # beta.star
#beta.star = matrix(c(-3, 1, 2, 3),nrow = d.X+1) # beta.star
n1 = 300 # sample size of the 1st data set.
n2 = 500 # 2nd
n3 = 1000 # 3rd
n = 50
sim=1
set.seed(sim)
X.rf = MASS::mvrnorm(n = n, mu, Sigma)
X.m1 = MASS::mvrnorm(n = n1, mu, Sigma) # Generate the covariates.
X.m1.1 = cbind(rep(1, n1), X.m1) # Add a column of 1's to X.m1.
p.m1 = 1/(1+exp(-X.m1.1%*%beta.star)) # the vector of probabilities
Y.m1 = rbinom(n1, size=1, p.m1) # the Bernoulli responses
# print(p.m1[1])
# print(mean(Y.m1))
```

```
# print(mean(p.m1))
# Generate data set 2. m1 means model 2.
X.m2 = MASS::mvrnorm(n = n2, mu, Sigma)
X.m2.1 = cbind(rep(1, n2), X.m2)
p.m2 = 1/(1+exp(-X.m2.1%*%beta.star))
Y.m2 = rbinom(n2, size=1, p.m2)
X.m3 = MASS::mvrnorm(n = n3, mu, Sigma)
X.m3.1 = cbind(rep(1, n3), X.m3)
p.m3 = 1/(1+exp(-X.m3.1%*%beta.star))
Y.m3 = rbinom(n3, size=1, p.m3)
#####
### Create data sets in the format of data frame.
#####
data.m1 = data.frame(Y=Y.m1, X.m1)
data.m2 = data.frame(Y=Y.m2, X.m2)
data.m3 = data.frame(Y=Y.m3, X.m3)
#####
### Apply logistic regression with reduced models to the data sets
#####
logit.m1 <- glm(Y ~ X1 + X2, data = data.m1, family = "binomial")
# print(logit.m1)
if(logit.m1$converged == FALSE)
{
  print("glm for logit.m1 is not convergent.")
  next
}
logit.m2 <- glm(Y ~ X2 + X3, data = data.m2, family = "binomial")
# print(logit.m2)
if(logit.m2$converged == FALSE)
{
  print("glm for logit.m2 is not convergent.")
  next
}
logit.m3 <- glm(Y ~ X1 + X3, data = data.m3, family = "binomial")
# print(logit.m3)
if(logit.m3$converged == FALSE)
{
  print("glm for logit.m3 is not convergent.")
  next
}
#####
### Obtain the estimators of the parameters in the reduced models.
#####
theta.m1 = logit.m1$coefficients
theta.m2 = logit.m2$coefficients
theta.m3 = logit.m3$coefficients
#####
### Find the covariance matrix estimators for the reduced models
#####
#####
# Basic notations for inputs
#####
K = 3 # Number of data sets
A1 = c(1, 2) # index set A1, the indexes of the covariates of data set 1.
A2 = c(2, 3) # index set A2
A3 = c(1, 3) # index set A3
X.m1.used = cbind(rep(1, n1), X.m1[, A1, drop=FALSE])
```

```
X.m2.used = cbind(rep(1, n2), X.m2[, A2, drop=FALSE])
X.m3.used = cbind(rep(1, n3), X.m3[, A3, drop=FALSE])
# str(X.m1.used)
# str(X.m2.used)
# str(X.m3.used)
##### Find Sigma.m1
T.1 = matrix(rep(0, (length(A1)+1)^2), nrow=length(A1)+1)
T.2 = T.1
for (i in 1:n1)
{
a = as.vector(exp(-X.m1.used[i, , drop=FALSE]%*%theta.m1))
  T.1 = T.1 + (a/(1+a)^2) * (t(X.m1.used[i, , drop=FALSE])%*%X.m1.used[i, , drop=FALSE])
}
for (i in 1:n1)
{
  a = as.vector(1/( 1 + exp(-X.m1.used[i, , drop=FALSE]%*%theta.m1)))
  T.2 = T.2 + (Y.m1[i]-a)^2 * (t(X.m1.used[i, , drop=FALSE])%*%X.m1.used[i, , drop=FALSE])
}
Sigma.m1 = solve(T.1)%*%T.2%*%solve(T.1) # This is actually Sigma.m1.n1.
##### Find Sigma.m2
T.1 = matrix(rep(0, (length(A2)+1)^2), nrow=length(A2)+1)
T.2 = T.1
for (i in 1:n2)
{
  a = as.vector(exp(-X.m2.used[i, , drop=FALSE]%*%theta.m2))
  T.1 = T.1 + (a/(1+a)^2) * (t(X.m2.used[i, , drop=FALSE])%*%X.m2.used[i, , drop=FALSE])
}
for (i in 1:n2)
{
  a = as.vector(1/( 1 + exp(-X.m2.used[i, , drop=FALSE]%*%theta.m2)))
  T.2 = T.2 + (Y.m2[i]-a)^2 * (t(X.m2.used[i, , drop=FALSE])%*%X.m2.used[i, , drop=FALSE])
}
Sigma.m2 = solve(T.1)%*%T.2%*%solve(T.1)
##### Find Sigma.m3
T.1 = matrix(rep(0, (length(A3)+1)^2), nrow=length(A3)+1)
T.2 = T.1
for (i in 1:n3)
{
  a = as.vector(exp(-X.m3.used[i, , drop=FALSE]%*%theta.m3))
  T.1 = T.1 + (a/(1+a)^2) * (t(X.m3.used[i, , drop=FALSE])%*%X.m3.used[i, , drop=FALSE])
}
for (i in 1:n3)
{
  a = as.vector(1/( 1 + exp(-X.m3.used[i, , drop=FALSE]%*%theta.m3)))
  T.2 = T.2 + (Y.m3[i]-a)^2 * (t(X.m3.used[i, , drop=FALSE])%*%X.m3.used[i, , drop=FALSE])
}
Sigma.m3 = solve(T.1)%*%T.2%*%solve(T.1)
names(theta.m1)=c("(Intercept)","Age","Height")
names(theta.m2)=c("(Intercept)","Height", "Weight")
names(theta.m3)=c("(Intercept)","Age", "Weight")
study1 = list(Coeff=theta.m1,Covariance=Sigma.m1,Sample_size=n1)
study2 = list(Coeff=theta.m2,Covariance=Sigma.m2,Sample_size=n2)
study3 = list(Coeff=theta.m3,Covariance=Sigma.m3,Sample_size=n3)
studies = list(study1,study2,study3)
model = "logistic"
reference = cbind(rep(1,n), X.rf)
colnames(reference) = c("(Intercept)","Age","Height", "Weight")
```

```
result.same = GENMETA(studies, reference, model,
initial_val = c(-1.2, log(1.3), log(1.3), log(1.3)))
GENMETA.summary(result.same)
```

# Index