Package 'BFI'

September 1, 2023

Type Package

Title Bayesian Federated Inference

Version 0.6.4

Date 2023-07-16

Author Hassan Pazira [aut, cre],
Marianne A. Jonker [aut],
Anthony C.C. Coolen [aut]

Maintainer Hassan Pazira <hassan.pazira@radboudumc.nl>

Description Bayesian Federated Inference method combines data from different (medical) centers without sharing them. In this version of the package, the user can fit models specifying Gaussian and Binomial (Logistic) families.

License GPL (>= 2)

URL https://github.com/hassanpazira/BFI

Encoding UTF-8

Suggests knitr,
rmarkdown,
testthat (>= 3.0.0)

VignetteBuilder knitr

Depends R (>= 2.10)

LazyData true

Imports devtools

Config/testthat/edition 3

RoxygenNote 7.2.3

R topics documented:

BFI-package *Bayesian Federated Inference*

Description

Bayesian Federated Inference method combines inference results from different (medical) centers without sharing the data. In this version of the package, the user can fit models specifying Gaussian and Binomial (Logistic) families. The package will be updated with more models soon.

Details

Package:	BFI
Type:	Package
Version:	0.6.4
Date:	2023-07-16
License:	GPL (>=2)

`MAP.estimation` and `bfi` are the main functions. All other functions are utility functions.

Some examples are provided in the vignettes accompanying this package in order to show how the package can be applied to a real data.

Author(s)

Hassan Pazira, Marianne A. Jonker, Anthony C.C. Coolen
Maintainer: Hassan Pazira <hassan.pazira@radboudumc.nl>

References

Jonker M.A., Pazira H. and Coolen A.C.C. (2023). *Bayesian Federated Inference for Statistical Models*, *Statistics in Medicine*, Vol. 0(0), 0-0. <https://doi.org/10.48550/arXiv.2302.07677>

bfi *Bayesian Federated Inference*

Description

`bfi` function can be used (in central server) to combine inference results from separate data sets (without combining the data) to approximate what would have been inferred had the data sets been merged. For now the function can handle linear and logistic regression models, but more models will be available in the near future. bfi command

Usage

```
bfi(theta_hats = NULL, A_hats, Lambda, const_var = NULL,
    stratified = FALSE, nuisance = 1L)
```

Arguments

theta_hats	a list of L vectors of the maximum a posteriori (MAP) estimates of the model parameters in the L centers. These vectors must have equal dimensions. See 'Details'.
A_hats	a list of L curvature matrices for L centers. These matrices must have equal dimensions. See 'Details'.
Lambda	a list of $L + 1$ matrices. Each matrix used as the prior of the inverse variance-covariance matrix of Gaussian distribution. The first L matrices are for L local centers and the last one is the chosen variance-covariance matrix for the Gaussian prior of the (fictive) combined data set. These matrices must have equal dimensions. If all the $L + 1$ matrices are the same, this argument can be one matrix or a list of one matrix. See 'Details'.
const_var	a vector of L elements. Each element represents a specific feature of the corresponding center. There must be only one specific value or attribute for each center. This vector could be a numeric, characteristic or factor vector.
stratified	logical flag for performing the stratified analysis. If stratified=TRUE, the parameters selected in the nuisance argument can be different across centers. Default is stratified=FALSE. See 'Details'.
nuisance	a vector of one or two elements. It is usable if stratified=TRUE. For the binomial family the length of the vector should be one which refers to 'intercept', and for gaussian this vector can have maximum two elements which refer to 'intercept' and/or 'sigma2'. See 'Details'.

Details

bfi function implements the BFI approach described in the paper Jonker et. al. (2023) given in the references. The results gathered from different (L) centers are combined, and the BFI estimates of the model parameters and curvature matrix evaluated at that point are returned.

The result from each center must be obtained using the MAP.estimation function separately, and then all of these results (coming from different centers) should be compiled into a list to be used as an input of bfi(). The models in the different centers should be defined in exactly the same way; among others, exactly the same covariates should be included in the models. The parameter vectors should be defined exactly the same, so that the estimates theta_hat's and A_hat's (the output of the MAP.estimation function) are defined in the same way.

Note that, the order of the estimates in the lists theta_hats, A_hats and Lambda must be the same, so that in every list the elements at the ℓ position are all from the same center. For example, the first three entries of bfi() could be as follows:

bfi(theta_hats=theta_list, A_hats=A_list, Lambda= Lambda_list)

where

theta_list=list(obj$_1$$theta_hat, ..., obj$_\ell$$theta_hat, ..., obj$_L$$theta_hat),

A_list=list(obj$_1$$A_hat,..., obj$_\ell$$A_hat,..., obj$_L$$A_hat),

Lambda_list=list(obj$_1$$Lambda,..., obj$_\ell$$Lambda,..., obj$_L$$Lambda),

and

obj$_\ell$=MAP.estimation(y=y$_\ell$, X=X$_\ell$, Lambda=Lambda$_\ell$) # must be done at the center ℓ

where y$_\ell$, X$_\ell$ and Lambda$_\ell$ are the data and the prior of the inverse variance-covariance matrix for the ℓ center.

Details related to stratified=TRUE should be added!!!....

Details related to const_var should be added!!!....

Value

`bfi` returns a list containing the following components:

`theta_hat`	the p- or $(p+1)$-dimensional vector of estimates obtained by combining the inference results from the L centers with the `'BFI'` methodology when `stratified=FALSE`, where p is the number of regression parameters including intercept. If `stratified=TRUE`, dimension of this vector is $L+p-1$ (for `binomial` family, or for `gaussian` family when `nuisance` has one element) or $2*L+p-1$ (for `gaussian` family when `nuisance` has two elements). See 'Details';
`A_hat`	the curvature matrix obtained by the `'BFI'` method when `stratified=FALSE`, such that it's a $(p\times p)$- or $(p+1)\times(p+1)$-dimensional matrix depending on the family used, where p is the number of regression parameters including intercept. If `stratified=TRUE`, it's a list of L matrices corresponding to each center. This is not the same as `A_hats` in 'Arguments'. See 'Details';
`sd`	the p- or $(p+1)$-dimensional vector of standard deviation of the estimates in `theta_hat` if `stratified=FALSE`, i.e., the vector equals `sqrt(diag(solve(A_hat)))`. If `stratified=TRUE`, it's a list of L vectors; one vector for every center. These vectors are standard deviation of parameter estimates obtained from the matrices in `A_hat`, i.e., the vector equals `sqrt(diag(solve(A_hat[[j]])))` where `j` refers to a center.

Author(s)

Hassan Pazira
Maintainer: Hassan Pazira <hassan.pazira@radboudumc.nl>

References

Jonker M.A., Pazira H. and Coolen A.C.C. (2023). *Bayesian Federated Inference for Statistical Models*. Statistics in Medicine, Vol. 0(0), 0-0. <https://doi.org/10.48550/arXiv.2302.07677>

See Also

`MAP.estimation`

Examples

```
#-------------
# y ~ Binomial
#-------------
set.seed(112358)
p <- 4 # number of regression coefficients (including intercept), is the same for all L=2 locations

##---------------
## Local center 1
##---------------
n1   <- 30
X1   <- data.frame(x1=rnorm(n1),                    # standard normal variable
                   x2=sample(0:1, n1, replace=TRUE), # dichotomous variable
                   x3=sample(1:3, n1, replace=TRUE)) # categorical variable
#true_beta <- rep(2, p)           # with an intercept b0=b1=b2=b3=2
eta1 <- 2 + apply(2 * X1, 1, sum) # linear predictor: X%*%\beta=\eta
mu1  <- binomial()$linkinv(eta1)  # inverse of the link function: g^{-1}(\eta)=\mu
y1   <- rbinom(n1, 1, mu1)
```

```
# we assume the same inverse covariance matrix for all locations
Lambda     <- inv.prior.cov(X1, 0.01, family=binomial)
fit1       <- MAP.estimation(y1, X1, family=binomial, Lambda)
theta_hat1 <- fit1$theta_hat # intercept and coefficient estimates
A_hat1     <- fit1$A_hat     # curvature matrix

##---------------
## Local center 2
##---------------
n2         <- 50
X2         <- data.frame(x1=rnorm(n2),                    # standard normal variable
                         x2=sample(0:1, n2, replace=TRUE), # dichotomous variable
                         x3=sample(1:3, n2, replace=TRUE)) # categorical variable
eta2       <- 2 + apply(2 * X2, 1, sum) # linear predictor: X%*%\beta=\eta
mu2        <- binomial()$linkinv(eta2)  # inverse of the link function: g^{-1}(\eta)=\mu
y2         <- rbinom(n2, 1, mu2)
fit2       <- MAP.estimation(y2, X2, family=binomial, Lambda)
theta_hat2 <- fit2$theta_hat
A_hat2     <- fit2$A_hat

###----------------------------
### Bayesian Federated Inference
###----------------------------
A_hats     <- list(A_hat1, A_hat2)
theta_hats <- list(theta_hat1, theta_hat2)
bfi(theta_hats, A_hats, Lambda)
```

`inv.prior.cov` *Creates an inverse covariance matrix for a Gaussian prior*

Description

`inv.prior.cov` builds a diagonal inverse covariance matrix for the Gaussian prior distribution based on the design matrix of covariates, that takes into account the number of regression parameters in case of categorical covariates. In case of a linear model, it also includes the variance of the measurement errors.

Usage

```
inv.prior.cov(X, lambda = 1, family = gaussian, intercept = TRUE,
              independ = TRUE, set_seed = NULL)
```

Arguments

`X`	design matrix of dimension $n \times p$, where p is the number of covariates/predictors without intercept.
`lambda`	the tuning parameter for ridge (L2) penalization that controls the strength of the ridge penalty. It could be either a single positive number, a vector of two elements, or a vector with length equal to p, $p+1$ or $p+2$ (where p is the number of covariates, i.e., number of columns of `X` argument) depends on the arguments `X`, `family`, and `intercept`. If `lambda` chosen by the user is a single positive number, the function `inv.prior.cov()` consider `lambda` as a vector whose all

elements are equal to that scalar value. Default is `lambda=1`. If the user choose a vector of only two elements as an entry value, the `inv.prior.cov()` function set `lambda` as a vector whose all elements, except the last one, are equal to the first element of the entry, and the last element is set to the second element of the entry. Used when `independ=TRUE`.

`family` a description of the error distribution and link function used to specify the model. This can be a character string naming a family function or the result of a call to a family function (see `family` for details). By default the `gaussian` family (with identity link function) is used.

`intercept` logical flag for fitting an intercept. The intercept is fitted if `intercept=TRUE` (the default) or set to zero if `intercept=FALSE`. If `intercept=FALSE`, the first row and column of this matrix are set to zero

`independ` logical flag for creating the prior covariance matrix with (in)dependent variables/parameters. If `TRUE` (the default), the result is a diagonal matrix which means the parameters are independent. If `FALSE`, the off-diagonal elements of the resulted matrix are generated from the standard normal distribution `rnorm()`, and the diagonal elements are the vector `lambda`.

`set_seed` if `independ=FALSE`, this argument (which should be an integer) ensures consistent results. If `set_seed=NULL` (the default), the results are not consistent.

Details

`inv.prior.cov` creates a matrix whose dimension depends on the arguments `X`, `family`, and `intercept`. If we assume the parameters are independent (`independ=TRUE`), the function `inv.prior.cov` returns a diagonal matrix with the vector `lambda` as its diagonal. If `independ=FALSE`, `inv.prior.cov` returns a matrix which is equal to `z%*%t(z)` where z is a matrix such that all elements are generated by standard normal distribution.

The output of `inv.prior.cov()` is used in the main functions `MAP.estimation()` and `bfi()`.

Value

`inv.prior.cov` returns a matrix. The dimension of the matrix depends on the arguments `X`, `family`, and `intercept`.

Author(s)

Hassan Pazira
Maintainer: Hassan Pazira <hassan.pazira@radboudumc.nl>

References

Jonker M.A., Pazira H. and Coolen A.C.C. (2023). *Bayesian Federated Inference for Statistical Models*, Statistics in Medicine, Vol. 0(0), 0-0. <https://doi.org/10.48550/arXiv.2302.07677>

See Also

`MAP.estimation`

Examples

```
#-------------
# y ~ Binomial
#-------------
X       <- data.frame(matrix(rnorm(50 * 4), 50, 4))
lambda  <- 0.05
# We assume the same (inverse) covariance matrix for all
# locations and consider independency of prior parameters
(Lambda <- inv.prior.cov(X, lambda, family=binomial))
# No intercept
(Lambda <- inv.prior.cov(X, lambda, family="binomial", intercept = FALSE))

#-------------
# y ~ Gaussian
#-------------
X       <- data.frame(matrix(rnorm(50 * 3), 50, 3))
lambda  <- 0.01
sigma2e <- 0.5
# If we consider dependency for all prior parameters:
(Lambda <- inv.prior.cov(X, family="gaussian", independ = FALSE, set_seed=1123))
# No intercept
(Lambda <- inv.prior.cov(X, family="gaussian", intercept = FALSE, independ = FALSE, set_seed=1123))
```

`MAP.estimation` *Maximum A Posteriori estimation*

Description

`MAP.estimation` function is used (in local centers) to estimate Maximum A Posterior (MAP) of the parameters for the GLM and Survival models.

Usage

```
MAP.estimation(y, X, family = gaussian, Lambda, intercept = TRUE,
               initial = NULL, control = list())
```

Arguments

`y`	response vector. If the `binomial` family is used, this argument can be a vector with entries 0 (failure) or 1 (success). Alternatively, the response can be a matrix where the first column is the number of "successes" and the second column is the number of "failures".
`X`	design matrix of dimension $n \times p$, where p is the number of covariates (predictors) plus intercept.
`family`	a description of the error distribution and link function used to specify the model. This can be a character string naming a family function or the result of a call to a family function (see `family` for details). By default the `gaussian` family (with identity link function) is used.
`Lambda`	the matrix used as the prior of the inverse variance-covariance matrix of Gaussian distribution. It can be created by the function `inv.prior.cov()`.

`intercept`	logical flag for fitting an intercept. The intercept is fitted if `intercept=TRUE` (the default) or set to zero if `intercept=FALSE`. Although the intercept is included in the model by default, it can be penalized.
`initial`	a vector specifying initial values for the parameters (intercept, coefficients and/or error variance) to be optimized over. For the `gaussian` family, it should be a $p+1$-dimensional vector with a non-negative value for the last element, and for `binomial` the length of the vector should be p, where p is the number of covariates plus intercept. Since the `'L-BFGS-B'` method is used in the algorithm, these values should always be finite. Default is a vector of zeros.
`control`	a list of control parameters. See 'Details'.

Details

`MAP.estimation` function finds the Maximum A Posteriori (MAP) estimates of model parameters, i.e., the values associated with the peak of the log-posterior distribution (the posterior mode). In other words, `MAP.estimation()` optimizes the log-posterior density with respect to the parameter vector to obtain its MAP estimation. In addition to the model parameters, i.e., coefficients (β's) and variance error (σ_e^2), the curvature matrix (Hessian of the log-posterior) is estimated around the mode.

The current version of the **BFI** package supports two families of GLM: `gaussian` and `binomial`. For the `gaussian` family, `MAP.estimation` function returns the $(p+1)$-dimensional vector of the MAP estimates of the coefficients $(\beta_0, \beta_1, \ldots, \beta_{p-1})$ and variance error parameter, respectively. When the `binomial` family is used, it returns the MAP estimates of the p coefficients while the variance error or dispersion is set to one.

Note that: although an intercept is included in the model by default (except for `family=cox`) using `intercept=TRUE`, it can be penalized. It can also be set to zero using `intercept=FALSE`.

Since all elements of the lists in the first three arguments of the `bfi` function must have equal dimensions, the function `MAP.estimation` creates the output for all centers with similar dimensions even if in some centers the intercept is set to zero `intercept=FALSE` (of course for the same `family`'s) by setting the corresponding elements to zero.

The `MAP.estimation` function returns an object of class '`bfi`'. Therefore, `summary()` can be used for the object returned by `MAP.estimation()`.

To solve unconstrained and bound-constrained optimization problems, the `MAP.estimation` function utilizes an optimization algorithm called Limited-memory Broyden-Fletcher-Goldfarb-Shanno with Bound Constraints (L-BFGS-B), Byrd et. al. (1995). The L-BFGS-B algorithm is a limited-memory "quasi-Newton" method that iteratively updates the parameter estimates by approximating the inverse Hessian matrix using gradient information from the history of previous iterations. This approach allows the algorithm to approximate the curvature of the posterior distribution and efficiently search for the optimal solution, which makes it computationally efficient for problems with a large number of variables.

By default, the algorithm uses a relative change in the objective function as the convergence criterion. When the change in the objective function between iterations falls below a certain threshold ('`factr`') the algorithm is considered to have converged.

In Jonker et. al. (2023), the Gaussian and non-informative priors were used on the parameters. In this case, the posterior distribution is highly expected to be unimodal for a Gaussian GLM, which means it has only one minimum and no local optima. This property ensures that the optimization algorithm will converge to the global minimum, guaranteeing convergence. After fitting the model, the convergence status can be examined by checking the `convergence` attribute of the model object using the `$convergence` attribute. See 'Value' for the possible options of this attribute.

If a logistic regression model (`family=binomial`) is used, it's generally more likely that the posterior distribution will be unimodal with these two priors, especially if the data provides strong information about the parameters or sample size is large. Since the starting point has a great impact on the converged point especially for densities which are not unimodal, multiple starting points can be used in the `initial` argument to check for unimodality. Indeed, the negative log-posterior density is minimized from an chosen starting point `initial`.

However, there could be cases with small data sets (non-informative data), high-dimensionality, or inappropriate optimization settings, which might lead to multimodality or other complex behavior in the posterior distribution, resulting in users experiencing convergence issues. In such cases, it may be necessary to investigate and adjust optimization parameters to facilitate convergence. It can be done using the `control` argument.

The argument `control` is a list that can supply any of the following components:

`maxit`: is the maximum number of iterations. Default is 1e2;

`factr`: controls the convergence of the `'L-BFGS-B'` method. Convergence occurs when the reduction in the objective is within this factor of the machine tolerance. Default is 1e7, that is a tolerance of about 1e-8;

`pgtol`: helps control the convergence of the `'L-BFGS-B'` method. It is a tolerance on the projected gradient in the current search direction. Default is zero, when the check is suppressed;

`trace`: is a non-negative integer. If positive, tracing information on the progress of the optimization is produced. Higher values may produce more tracing information: for the method `'L-BFGS-B'` there are six levels of tracing. To understand exactly what these do see the source code of `optim` function in the **stats** package;

`REPORT`: is the frequency of reports for the `'L-BFGS-B'` method if `'control$trace'` is positive. Default is every 10 iterations;

`lmm`: is an integer giving the number of `BFGS` updates retained in the `'L-BFGS-B'` method. Default is 5.

Value

`MAP.estimation` returns a list containing the following components:

`theta_hat`	the p- or $(p+1)$-dimensional vector corresponding to the maximum a posteriori (MAP) estimation of the parameters, where p is the number of covariates plus intercept. If `intercept=FALSE`, the first element of the vector is set to zero;
`A_hat`	the curvature matrix around the point `theta_hat`. It's a $(p \times p)$- or $(p+1) \times (p+1)$-dimensional matrix depending on the family used. If `intercept=FALSE`, the first row and column of the matrix are set to zero;
`sd`	the p- or $(p+1)$-dimensional vector of standard deviation of estimates in `theta_hat`, i.e., `sqrt(diag(solve(A_hat)))`. If `intercept=FALSE`, the first element of this vector is set to zero;
`Lambda`	the matrix used as the prior of the inverse variance-covariance matrix. If `intercept=FALSE`, the first row and column of this matrix are set to zero;
`formula`	the formula of the model;
`n`	sample size;
`np`	the number of coefficients/regression parameters. In other words, number of predictors plus the intercept if `intercept=TRUE`, or without intercept if `intercept=FALSE`;
`value`	the value of the 'negative' log-likelihood function corresponding to `theta_hat`;
`family`	a description of the error distribution used in the model;

`intercept`	logical flag used to fit an intercept if `TRUE`, or set to zero if `FALSE`;
`convergence`	an integer value used to encode the warnings and the errors related to the algorithm used to fit the model. The values returned are: **0** algorithm has converged; **1** maximum number of iterations ('`maxit`') has been reached; **2** Warning from the 'L-BFGS-B' method. See the message after this value;
`control`	the list of control parameters used to compute the MAP estimates.

Author(s)

Hassan Pazira
Maintainer: Hassan Pazira <hassan.pazira@radboudumc.nl>

References

Jonker M.A., Pazira H. and Coolen A.C.C. (2023). *Bayesian Federated Inference for Statistical Models*. Statistics in Medicine, Vol. 0(0), 0-0. <https://doi.org/10.48550/arXiv.2302.07677>

Byrd R.H., Lu P., Nocedal J. and Zhu C. (1995). *A limited memory algorithm for bound constrained optimization*. SIAM Journal on Scientific Computing, 16, 1190-1208. <https://doi.org/10.1137/0916069>

See Also

`bfi` and `summary.bfi`

Examples

```
#-------------
# y ~ Gaussian
#-------------

set.seed(11235813)
n      <- 30
p      <- 3    # number of coefficients (without intercept)
X      <- data.frame(matrix(rnorm(n * p), n, p))
eta    <- 1 + 2 * X[,1]    # with an intercept
mu     <- gaussian()$linkinv(eta)
sigma2 <- 1.5
# the true theta is c(1, 2, 0, 0, sigma2)
y      <- rnorm(n, mu, sd = sqrt(sigma2))
lambda <- 0.01
# inverse of covariance matrix:
Lambda <- inv.prior.cov(X, lambda = c(lambda,sigma2), family = gaussian)

# MAP estimates of the parameters of interest (including 'intercept') and curvature matrix
(fit <- MAP.estimation(y, X, family = gaussian, Lambda))
class(fit)

# MAP estimates without 'intercept'
Lambda <- inv.prior.cov(X, lambda = c(lambda,sigma2), family = gaussian,  intercept = FALSE)
(fit1 <- MAP.estimation(y, X, family = gaussian, Lambda, intercept = FALSE))
```

Nurses *Nurses' stress in different hospitals*

Description

This data set contains three-level simulated data from a hypothetical study on stress in hospitals. The data are from nurses working in wards nested within hospitals. It is a cluster-randomized experiment. In each of 25 hospitals, four wards are selected and randomly assigned to an experimental and a control condition. In the experimental condition, a training program is offered to all nurses to cope with job-related stress. After the program is completed, a sample of about 10 nurses from each ward is given a test that measures job-related stress. Additional variables are: nurse age (years), nurse experience (years), nurse gender (`0` = `male`, `1` = `female`), type of ward (`0` = `general care`, `1` = `special care`), and hospital size (`0` = `small`, `1` = `medium`, `2` = `large`).

Usage

```
data(Nurses)
```

Source

https://multilevel-analysis.sites.uu.nl/datasets/

References

Hox, J., Moerbeek, M., and van de Schoot, R. (2010). *Multilevel Analysis: Techniques and Applications*, Second Edition (2nd ed.). *Routledge*. <https://doi.org/10.4324/9780203852279>

summary.bfi *Summarizing BFI Fits*

Description

Summary method for an object with class 'bfi' created by the `MAP.estimation` function.

Usage

```
## S3 method for class 'bfi'
summary(object, curmat = FALSE,
        digits = max(3, getOption("digits") - 3), ...)
```

Arguments

`object`	fitted `bfi` object.
`curmat`	logical; if `TRUE`, the curvature matrix around the estimated parameters is returned and printed. Default is `FALSE`.
`digits`	significant digits in printout.
`...`	additional arguments affecting the summary produced.

Details

summary.bfi() gives information about the MAP estimates of parameters of the model. It can be used for the bfi objects built by the MAP.estimation function.

The output of the summary method shows the details of the model, i.e. formula, family and link function used to specify the generalized linear model, followed by information about the estimates, standard deviations and confidence intervals. Information about the log-likelihood and convergence status are also provided.

For a model fitted without an intercept (i.e., in MAP.estimation() the argument is intercept=FALSE), summary.bfi() exhibits the outputs without the intercept unlike the outputs of MAP.estimation() which are replaced by zeros.

By default, summary.bfi function does not return the curvature matrix, but the user can use curmat=TRUE to print it.

Value

summary.bfi returns an object of class summary.bfi, a list with the following components:

theta_hat	the component from object. The last element of this vector is the estimate of the dispersion parameter (sigma2). See the MAP.estimation function.
A_hat	the component from object. See the MAP.estimation function.
sd	the component from object. The last element of this vector is the square root of the estimated dispersion. See the MAP.estimation function.
Lambda	the component from object. See the MAP.estimation function.
formula	the component from object. See the MAP.estimation function.
n	the component from object. See the MAP.estimation function.
np	the component from object. See the MAP.estimation function.
value	the component from object. See the MAP.estimation function.
family	the component from object. See the MAP.estimation function.
intercept	the component from object. See the MAP.estimation function.
convergence	the component from object. See the MAP.estimation function.
control	the component from object. See the MAP.estimation function.
Estimate	the coefficients without the estimate sigma2. Moreover, if the intercept was not fitted (i.e., object$intercept is FALSE), (Intercept) is not shown here, while it can be seen in object$theta_hat which is zero. See 'Examples'.
se	the standard error. Here se is equal to the standard deviation (sd).
CurMat	the curvature matrix. If the intercept was not fitted (i.e., object$intercept is FALSE), the row and column related to the intercept will not be shown here unlike object$A_hat. See 'Examples'.
logLik	the value of the loglikelihood function corresponding to estimates (theta_hat). This is the minus of the value component.
link	the link function. By default the gaussian family with identity link function and the binomial family with logit link function are used.
dispersion	the estimated variance of the random error, i.e., sigma2. The dispersion is taken as 1 for the binomial family.
CI	a 95% confidence interval.

Author(s)

Hassan Pazira
Maintainer: Hassan Pazira <hassan.pazira@radboudumc.nl>

See Also

MAP.estimation and bfi

Examples

```
#------------------------------
# y ~ Gaussian (with intercept)
#------------------------------

set.seed(1123581)
n       <- 30
p       <- 4
X       <- data.frame(matrix(rnorm(n * p), n, p))
b       <- 1:2
eta     <- b[1] + X[, 1] * b[2]
mu      <- gaussian()$linkinv(eta)
sigma2e <- 0.5
y       <- rnorm(n, mu, sd=sqrt(sigma2e))
lambda  <- 0.1
Lambda  <- inv.prior.cov(X, lambda=c(lambda,sigma2e), family=gaussian)
fit     <- MAP.estimation(y, X, family=gaussian, Lambda)
class(fit)

summary(fit)
sumfit <- summary(fit, curmat = TRUE)
sumfit$logLik
sumfit$dispersion
sumfit$CI
class(sumfit)


#---------------------------------
# y ~ Binomial (without intercept)
#---------------------------------

n0       <- 40
p0       <- 5
X0       <- data.frame(matrix(rnorm(n0 * p0), n0, p0))
eta0     <- 1 + X0[, 1] * 2
mu0      <- binomial()$linkinv(eta0)
y0       <- rbinom(n0, 1, mu0)
Lambda0  <- inv.prior.cov(X0, lambda=0.5, family=binomial, intercept=FALSE)
fit0     <- MAP.estimation(y0, X0, family=binomial, Lambda0, intercept=FALSE)

sumfit0 <- summary(fit0, curmat = TRUE)
sumfit0$A_hat
sumfit0$theta_hat[-length(sumfit0$theta_hat)]
sumfit0$Estimate
```

trauma *Trauma patients from different hospitals*

Description

This data set consists of data of 371 trauma patients from three hospitals. The binary variable `mortality` is used as an outcome, and variables `age`, `sex`, the Injury Severity Score (`ISS`, ranging from 1 (low) to 75 (high)) and the Glasgow Coma Scale (`GCS`, which expresses the level of consciousness, ranging from 3 (low) to 15 (high)) are used as covariates. The data originate from multiple hospitals which can be categorized in three groups as: peripheral hospital without a neuro-surgical unit (`Status = 1`), peripheral hospital with a neuro-surgical unit (`Status = 2`), and academic medical center (`Status = 3`).

Usage

```
data(trauma)
```

References

Jonker M.A., Pazira H. and Coolen A.C.C. (2023). *Bayesian Federated Inference for Statistical Models*, *Statistics in Medicine*, Vol. 0(0), 0-0. <https://doi.org/10.48550/arXiv.2302.07677>

Draaisma J.M.Th, de Haan A.F.J., Goris R.J.A. (1989). *Preventable Trauma Deaths in the Netherlands - A prospective Multicentre Study*, *The journal of Trauma*, Vol. 29(11), 1552-1557.

Index