Package 'BayHap'

February 19, 2015

Type Package

Title Bayesian analysis of haplotype association using Markov Chain Monte Carlo

Version 1.0.1

Date 2013-03-13

Author Raquel Iniesta and Victor Moreno

Maintainer Raquel Iniesta <riniesta@pssjd.org>

Description The package BayHap performs simultaneous estimation of uncertain haplotype frequencies and association with haplotypes based on generalized linear models for quantitative, binary and survival traits. Bayesian statistics and Markov Chain Monte Carlo techniques are the theoretical framework for the methods of estimation included in this package. Prior values for model parameters can be included by the user. Convergence diagnostics and statistical and graphical analysis of the sampling output can be also carried out.

Depends boa

License GPL (>= 2)

NeedsCompilation yes

Repository CRAN

Date/Publication 2013-03-13 21:25:31

R topics documented:

BayHap-package *Bayesian analysis of haplotype association using MCMC*

Description

The package BayHap implements the simultaneous estimation of uncertain haplotype frequencies and the association with haplotypes based on generalized linear models for quantitative, binary and survival traits. Bayesian statistics and Markov Chain Monte Carlo techniques are the theoretical framework for the methods of estimation included in this package. Prior values for model parameters can be specified by the user. Convergence diagnostics and statistical and graphical analysis of the sampling output can be also carried out.

Details

Package: BayHap Type: Package Version: 1.0 Date: 2010-11-01 License: GPL (>= 2)

The main function of this package is the `bayhapReg` function. Given a sample of genotypes, this function carries out simultaneous estimations of haplotype frequences and estimations of the parameters of a chosen generalized linear model with the haplotype variable as a risk factor. Quantitative, binary and survival traits are handled by this function and modeled through linear regression, logistic regression and weibull regression models. Terms of interaction can be included in these models. Three different inheritance models are alowed in the analysis: `additive`,`dominant` or `recessive`. Bayesian statistics and Markov Chain Monte Carlo techniques are the theoretical framework for the methods of estimation included in this package. Samples of posterior distributions for parameters are generated through Random Walk, Slice Sampler and Gibbs Sampler techniques. To capture the uncertainty of the haplotypical sample, frequencies and model parameters are estimated simultaneously, so at each step of the sampling of model parameters, one pair of haplotypes is also sampled for every individual with uncertain haplotypes using the previous frequencies in the chain. Prior values for these parameters can be also included by the user. Once samples are drawn, convergence diagnostics and statistical and graphical analysis of the sampling output must be carried out to ensure the convergence of the chain. First of all, users have to run the `setupData` function to have the data.frame object prepared to be inserted in `bayhapReg`. Look at the examples below, where are considered two snps, one covariate and a quantitative response. In the example is shown a linear regression with an interaction term, with and without prior information. Note that in case of include prior information, before the execution of `bayhapReg` is necessary to run the `bayhapFreq` function in order to get the labels for each haplotype that occurred in the genotypes sample. From this resulting object of class `freq` the haplotype names needed in `haplo.prior` function can be extracted. Then, the names of the haplotypes chosen by the user to have prior information, prior values of their model coefficients and its standard deviation must be also introduced in `haplo.prior` function, all

of them respectively in the same order. Take care of not put prior information to haplotypes with frequency below the treshold specified in `freq.min`. This is not allowed by the program since rares category do not accept prior information. Beyond all of this, the analysis can be done without prior information.

Once bayhapReg has been executed use the print method to see the resulting estimations. To evaluate the convergence of the method and so, the validity of the results, further diagnose analysis must be done. Plot running mean `plotRmean` to test the stability of the running mean of chain values regards to the burnin and total number of iterations chosen, plot autocorrelations `plotACF` to check the decrease as the chain go forward, plot the density `plotDensity` to check the posterior distribution, plot the trace histories `plotTrace` and execute `conv.test` to check different test of convergence. If the chain does not converge, is recommended to perform some tunning as modifying the burnins values and the total number of iterations. If several models are adjusted, you can choose the better by using the BIC criterion. The lower the BIC value, the better the model fit.

Author(s)

Raquel Iniesta and Victor Moreno Maintainer: Raquel Iniesta <riniesta@pssjd.org>

References

Iniesta R and Moreno V (2008) Assessment of Genetic Association using Haplotypes Inferred with Uncertainty via Markov Chain Monte Carlo. Monte Carlo and Quasi-Monte Carlo Methods, Springer Berlin Heidelberg 529-535.

See Also

`setupData`, `bayhapFreq`, `bayhapReg`, `haplo.prior`, `print.reg`, `print.freq`, `plotACF`, `plotTrace`, `plotDensity`, `plotRmean`

Examples

```
snp1<-c("C/C", "C/C", "C/C", "C/C", "C/C", "C/C", "C/C", "C/C",
"C/C","C/T", "C/C", "C/C", "C/C", "C/C", "C/C", "C/C", "C/C",
"C/C","C/C", "C/C", "C/C", "C/T", "C/C", "C/C", "C/C", "C/C",
"C/C","C/C", "C/C", "C/C", "C/C", "C/C", "C/C", "C/C", "C/C",
"C/C","C/C", "C/C", "C/C", "C/C", "C/C", "C/C", "C/C", "C/C",
"C/C","C/C", "C/C", "C/C", "C/C", "C/C")

snp2<-c("G/G", "G/G", "G/G", "G/G", "G/G", "G/G", "G/G", "G/G",
"G/G","G/G", "G/G", "G/G", "G/G", "A/G", "G/G", "G/G", "G/G",
"G/G","G/G", "G/G", "G/G", "G/G", "G/G", "G/G", "A/G", "G/G",
"G/G","G/G", "G/G", "G/G", "G/G", "G/G", "A/G", "G/G", "G/G",
"G/G","G/G", "G/G", "G/G", "G/G", "G/G", "G/G", "G/G", "G/G",
"G/G","G/G", "G/G", "G/G", "G/G", "G/G")

covariate<-c(1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1,
1, 1, 1, 0, 0, 0, 0, 0, 0)

response.q<-c(-0.3566, 0.4999, 0.9204, 0.8139, 1.8662, -0.4734, -0.7104,
-1.5904, -0.4367, 1.4043, 0.2683, 0.2324, 0.0217, 0.527, -2.2662,
```

```
0.1757, 0.378, 0.5139, -0.5167, -0.6913, 0.3385, -0.1095, -1.293,
1.7752, -1.4137, -0.502, -0.2889, -0.9809, -1.2051, -0.4104,
0.1964, -0.5435, 1.7636, 0.6596, 0.767, -0.4716, 1.1389, 0.8475,
-0.0428, 0.5691, 1.4069, -0.7324, 1.8495, -1.4328, 0.8782, -0.2168,
-0.006, 0.0517, 0.5135, 0.7965)

data.orig<-data.frame(snp1,snp2,covariate,response.q)
data<-setupData(data.orig,snps.name=c("snp1","snp2"),sep="/")


res.q<-bayhapReg(formula=response.q~haplotypes+haplotypes*covariate,data=data,
                 family="gaussian",t.model="additive",na.snp.action="keep",
                 freqmin=0.01,burn.in.haplo=5000,burn.in.pheno=5000,
                 total.iter=5000,devhaplo=0.1,dist=1,lag.number=10,
                 sign=0.05,file=TRUE,prior.val=haplo.prior(),verbose=2)

print(res.q)

#In case of having prior information, first run bayhapFreq

res.freq<-bayhapFreq(data=data,na.snp.action="keep",col.snps=1:2,sep="/",
                     total.iter.haplo=5000)

#The next table is the result of bayhapFreq:

#Haplotypes snp1 snp2    Freq Std.error     ICL     ICU
#    haplo.1    T    G 0.02473   0.01282 0.00297 0.05328
#    haplo.2    C    A 0.03602   0.01437 0.00624 0.06949
#    haplo.3    T    A 0.00000   0.00001 0.00000 0.00582
#    haplo.4    C    G 0.93924   0.02010 0.89985 0.98457

#if the user has prior information for haplotypes TG and CA
#(in a linear model, these are prior values for the differences
#and the standard deviation), then haplos.names must be "haplo.1" and "haplo.2",
#as shown in the next call:

res.q.prior<-bayhapReg(formula=response.q~haplotypes+haplotypes*covariate,data=data,
                       family="gaussian",t.model="additive",na.snp.action="keep",
                       freqmin=0.01,burn.in.haplo=5000,burn.in.pheno=5000,
                       total.iter=5000,devhaplo=0.1,dist=1,lag.number=10,sign=0.05,
                       file=TRUE,prior.val=haplo.prior(res=res.freq,
                       haplos.name=c("haplo.1","haplo.2"),
                       coeff=c(1.1,0.8),sd.coeff=c(0.9,0.5)),verbose=2)

print(res.q.prior)

#To check the results, run plot functions and:

correl(res.q)
conv.test(res.q)

#To have an indicator of how well the model fits the data
```

```
BIC(res.q)
```

`autocorr` *Autocorrelation Function*

Description

Computes lag autocorrelations for the parameters in an MCMC sequence.

Usage

```
autocorr(x, lag, keep.rares.autocorr = FALSE)
```

Arguments

`x`	An object of class `reg` or `freq`.
`lag`	Numerical vector of lags where autocorrelation must be estimated.
`keep.rares.autocorr`	Logical TRUE or FALSE indicating whether autocorrelation for rares category must be shown or not.

Value

A matrix whose columns and rows contain the estimated autocorrelation functions at the specified lags and the monitored parameters, respectively.

Author(s)

Original version by Brian J. Smith in package Boa.

Adapted version by Raquel Iniesta <riniesta@pssjd.org>

See Also

`correl`, `plotACF`

bayhapFreq *Estimation of haplotype frequencies*

Description

This function performs the estimation of uncertain haplotype frequencies. Bayesian statistics and Markov Chain Monte Carlo techniques are the theoretical framework for the method implemented in this function. A random walk sampling draws the posterior distribution for haplotype frequencies. Convergence diagnostics and statistical and graphical analysis of the sampling output must be carried out.

This function calls an external .c function named `mcmc`.

`print` displays tables showing the resulting estimation for the parameters in the MCMC sequence with their standard deviation and credibility interval. The argument of this function is an object of class `freq`.

Usage

```
bayhapFreq(data, na.snp.action = "omit", snps.name = NULL,
           col.snps = NULL, sep = "/", total.iter.haplo = 10000,
           devhaplo = 0.1, dist = 1, lag.number = 10,
           sign = 0.05, file = FALSE, verbose = 2)
```

Arguments

`data`	A data.frame in which rows represent each individual and each SNP is represented by a column. For each individual, SNPs have to be a character string holding both alleles with the separator specified by the argument `sep`.
`na.snp.action`	Action to do with genotype missing data. The default value `omit` will exclude of the analysis individuals with missing value in at least one SNP.
`snps.name`	A character vector with data columns names for SNPs. If this value is the default NULL, col.snps must be reported.
`col.snps`	A numerical vector indicating the positions of the columns in data where SNPs are stored. If this value is the default NULL, snps.name must be reported.
`sep`	Character separator to divide alleles.
`total.iter.haplo`	Total number of iterations used to compute the total avarage of the chain. The default should be modified when convergence fails.
`devhaplo`	Deviation used in random walk sampling during the generation of the markov chain for haplotype frequency. This value have to be increased when convergence to local minima.
`dist`	The distribution to sample from to have the value which determines the next step of the random walk sampling. Default value 1 represents the uniform distribution. For normal distribution choose 0.
`lag.number`	Periodical number of iterations discarded during the avarage period.

sign	Probability used to compute credibility intervals.
file	During the sampling period, a file called "outhaplo*.txt" is generated containing all values of each parameter chain to let the user make the posterior diagnose. If file=TRUE this file will be keept on your working directory after the use of this function. Otherwise if file is set as FALSE the file will be removed when execution finishes.
verbose	There are three levels of showing messages during the function execution: 0 (any message), 1 (some messages) and the default 2 (all possible messages).

Details

Run bayhapFreq to compute frequencies for all possible haplotypes in the sample without collapsing in rare category. This could be useful to decide the value for freq.min needed as an argument in bayhapReg.

The execution of this function can take several minutes depending on the number of individuals and the number of loci, so it is recommended setting the verbose argument as 2.

Further diagnose analysis must be done to evaluate the convergence of the method and so, the validity of the results. Plot running mean with plotRmean to test the stability of the running mean of chain values regards to the burnin and total number of iteration chosen, plot autocorrelation with plotACF to check how it decreases as the chain go forward, plot the density with plotDensity to check the posterior distribution, plot the trace histories plotTrace and run conv.test to check different test of convergence.

Value

The object of class freq returned is used for print methods. Print shows a table with lables for each haplotype, the estimated frequency, the standard error of these estimations and the credibility interval with the signification specified in sign.

Author(s)

Raquel Iniesta <riniesta@pssjd.org>

References

Iniesta R and Moreno V (2008) Assessment of Genetic Association using Haplotypes Inferred with Uncertainty via Markov Chain Monte Carlo. Monte Carlo and Quasi-Monte Carlo Methods, Springer Berlin Heidelberg 529-535.

Examples

```
snp1<-c("C/C", "C/C", "C/C", "C/C", "C/C", "C/C", "C/C", "C/C",
"C/C","C/T", "C/C", "C/C", "C/C", "C/C", "C/C", "C/C", "C/C",
"C/C","C/C", "C/C", "C/C", "C/T", "C/C", "C/C", "C/C", "C/C",
"C/C","C/C", "C/C", "C/C", "C/C", "C/C", "C/C", "C/C", "C/C",
"C/C","C/C", "C/C", "C/C", "C/C", "C/C", "C/C", "C/C", "C/C",
"C/C","C/C", "C/C", "C/C", "C/C", "C/C")

snp2<-c("G/G", "G/G", "G/G", "G/G", "G/G", "G/G", "G/G", "G/G",
```

```
"G/G","G/G", "G/G", "G/G", "G/G", "A/G", "G/G", "G/G", "G/G",
"G/G","G/G", "G/G", "G/G", "G/G", "G/G", "G/G", "A/G", "G/G",
"G/G","G/G", "G/G", "G/G", "G/G", "G/G", "A/G", "G/G", "G/G",
"G/G","G/G", "G/G", "G/G", "G/G", "G/G", "G/G", "G/G", "G/G",
"G/G","G/G", "G/G", "G/G", "G/G", "G/G")

data<-data.frame(snp1,snp2)

res.freq<-bayhapFreq(data=data,na.snp.action="keep",col.snps=1:2,
                     sep="/",total.iter.haplo=5000)

print(res.freq)
```

---

bayhapReg *Bayesian analysis of haplotype association using Markov Chain Monte Carlo*

Description

This is the main function of this pachage. This function implements the simultaneous estimation of uncertain haplotype frequencies and the association between haplotypes and quantitative, binary and survival traits based on generalized linear models. Prior values for model parameters can be specified by the user. Convergence diagnostics and statistical and graphical analysis of the sampling output must be carried out to ensure the convergence of the chain.

This function calls an external .c function named `mcmc`.

`print` displays tables showing the resulting estimation for the parameters in the MCMC sequence with their standard deviation and probability interval. The argument is an object of class `reg`.

Usage

```
bayhapReg(formula = formula, data, family = "gaussian",
          t.model = "additive", sep = "/",
          na.snp.action = "omit", freqmin = 0.01,
          burn.in.haplo = 10000, burn.in.pheno = 10000,
          total.iter = 10000, devhaplo = 0.1, dist = 1,
          lag.number = 10, sign = 0.05, file = FALSE,
          prior.val = haplo.prior(), verbose = 2)
```

Arguments

`formula` An expression of the form response~predictors. Terms of interaction have to be expressed as in the formula style of other regression models (see documentation for `formula`). Variables interacting with haplotypes must be also main effects in the model. Hence, if you use 'haplotypes:a' notation, you are bound to make appear the variable a in formula as 'response~haplotypes+a+haplotypes:a'. Otherwise you can use the compact form 'response~haplotypes+haplotypes*a'. The reference category for all categorical variables in the model is the first value in alphabetical order for character strings and the minor value in case of numerical variables.

data	The data.frame returned from setupData in which rows represent each individual and columns are the variables ocurring in the formula (response, SNPs information and covariates) plus an internal variable called haplotypes. Each SNP in the original data.frame has to be a character string holding both alleles with the separator specified by the argument sep. The word haplotypes is an internal word, so no variable in the original data.frame can bear this name.
family	One of the three possible models to perform: "gaussian", "binomial" or "survival".
t.model	The type of inheritance model to follow. There are three options: additive(default), dominant or recessive.
sep	Character separator to divide alleles.
na.snp.action	Action to do with genotype missing data. The default value omit will exclude of the analysis individuals with missing value in at least one SNP.
freqmin	Minimum treshold for the frequency of haplotype which have to be included in the model. Haplotyes with frequency below freqmin will be collapsed in a new category called rares. The default value of 0.01 is the minimum accepted.
burn.in.haplo	Number of initial oscilating iterations to be discarded of the markov chain of haplotype frequencies. The default should be modified when convergence fails in the chain.
burn.in.pheno	Number of initial iterations to be discarded of the markov chain for the coefficients of the model. The default should be modified when convergence fails in the chain.
total.iter	Total number of iterations added to burn.in.haplo and burn.in.pheno to compute the total avarage of both chains. The default should be modified when convergence fails for one or both chains.
devhaplo	Deviation used in Random Walk sampling during the generation of the markov chain for haplotype frequency. This value have to be increased when convergence to local minima.
dist	The distribution to sample from to have the value which determines the next step of the random walk sampling. Default value 1 represents the uniform distribution. For normal distribution choose 0.
lag.number	Periodical number of iterations discarded during the avarage period.
sign	Signification used to compute intervals of probability.
file	During the sampling period, a file called "outhaplo.txt" is generated containing all the values of each parameter chain to make the posterior diagnose. If file=TRUE this file will be keept on your working directory after the execution of the function. Otherwise if file is set as FALSE the file will be removed.
prior.val	The value returned by haplo.prior. Use it when you want to consider a Bayesian approach. See the help of haplo.prior for more details.
verbose	The three possible levels of showing messages during the function execution: 0 (any message), 1 (some messages) and the default 2 (all possible messages).

Details

To compute frequencies for all possible haplotypes in the sample without collapsing in rare category, run `bayhapFreq`. This also could be useful to decide the value for `freq.min` before running `bayhapReg`.

The execution of this function can take several minutes depending on the number of individuals, number of loci and the number of covariates and interactions, so it is recommended setting the verbose argument as 2.

Further diagnose analysis must be done to evaluate the convergence of the method and so, the validity of the results. Plot running mean with `plotRmean` to test the stability of the running mean of chain values regards to the burnin and total number of iteration chosen, plot autocorrelations with `plotACF` to check the decrease as the chain go forward, plot the density with `plotDensity` to check the posterior distribution and plot the trace histories with `plotTrace` (`plotReg` also do all these plots) Finally run `conv.test` to check different test of convergence.

The density function for a possible parametrization of the Weibull distribution with 'shape' parameter r and 'scale' parameter k is given by:

f(t) = kr[(rt)^(k-1)] exp(- (rt)^k)

For our purposes, these functions have been parametrized. The cumulative distribution function is:

F(t) = 1 - exp(-(t/b)^a)

where t is a non-negative value, the mean is E(t) = b Gamma(1 + 1/a), and the Var(t) = b^2 * (Gamma(1 + 2/a) - (Gamma(1 + 1/a))^2).

If covariates are considered, the density function could be parametrized as follows. Let be z the covariate, and `beta`the vector of the regression coefficients. We take $r^k=exp(beta*z)$ and so the density function is:

f(t)= k exp(zbeta) [t^(k-1)] exp(-(t^k) exp(zbeta))

For this parametrization, the hazard function is:

h(t)=k exp(zbeta)t^(k-1)

and the survival function is:

s(t)=exp(-(t^k) exp(zbeta))

Note that before include prior information is necessary to run the `bayhapFreq` function to get the labels for each haplotype that occurred in the genotypes sample. From this resulting object of class `freq` the haplotype names needed in `haplo.prior` function can be extracted. Then, the prior values of the mean of the coefficients for the haplotypes chosen by the user and its standard deviation must be also introduced in `haplo.prior` function in the same order of the corresponding haplotype name. It is, for each prior-informated haplotype, its coefficient in the regression model follows a normal distribution:

beta_haplotype~N(mean_haplotype,var_haplotype)

If family model is binomial, there are two possible options. You can put prior information for model coefficients (mean and standard deviation) or you can put prior information for the OR, and for the confidence interval and significance of each interval, instead of inform about the coefficients. See the help of `haplo.prior` for more specifications about the arguments.

Take care of not put prior information to haplotypes with frequency below the treshold specified in `freqmin`. This is not allowed by the program since rares category do not accept prior information.

Value

An object of class `reg` is returned for print methods. There are some fixed tables and values generated with `print.reg` for all models and other tables and values only given for an specific family model or case. The next tables and values are always shown by print method:

`Haplotype Frequencies`
: Table showing haplotypes, their estimated frequency, the standard error for the estimation and the probability interval.

`Coefficients`
: Table showing estimations of the regression coefficients. For linear and weibull models, estimates of variance and scale parameters are also returned.

`Formula`
: The considered formula.

`Model`
: The function link.

`BIC`
: Bayesian Information Criterion. The lower the value, the better the model fits.

In case of binomial family model, one table showing Odds Ratios is added.

If there are interactions in formula and the family is gaussian or binomial then three tables are also added to the output: the cross classification interaction table, the table for haplotypes within covariate, and the table for covariate within haplotypes.

Author(s)

Raquel Iniesta <riniesta@pssjd.org>

References

Iniesta R and Moreno V (2008) Assessment of Genetic Association using Haplotypes Inferred with Uncertainty via Markov Chain Monte Carlo. Monte Carlo and Quasi-Monte Carlo Methods, Springer Berlin Heidelberg 529-535.

See Also

`setupData`, `bayhapFreq`, `haplo.prior`, `plotReg`

Examples

```
##quantitative response (linear regression model) with prior values
##for some haplotype frequencies:

snp1<-c("C/C", "C/C", "C/C", "C/C", "C/C", "C/C", "C/C", "C/C",
"C/C","C/T", "C/C", "C/C", "C/C", "C/C", "C/C", "C/C", "C/C",
"C/C","C/C", "C/C", "C/C", "C/T", "C/C", "C/C", "C/C", "C/C",
"C/C","C/C", "C/C", "C/C", "C/C", "C/C", "C/C", "C/C", "C/C",
"C/C","C/C", "C/C", "C/C", "C/C", "C/C", "C/C", "C/C", "C/C",
"C/C","C/C", "C/C", "C/C", "C/C", "C/C")

snp2<-c("G/G", "G/G", "G/G", "G/G", "G/G", "G/G", "G/G", "G/G",
"G/G","G/G", "G/G", "G/G", "G/G", "A/G", "G/G", "G/G", "G/G",
"G/G","G/G", "G/G", "G/G", "G/G", "G/G", "G/G", "A/G", "G/G",
```

```
"G/G","G/G", "G/G", "G/G", "G/G", "G/G", "A/G", "G/G", "G/G",
"G/G","G/G", "G/G", "G/G", "G/G", "G/G", "G/G", "G/G", "G/G",
"G/G","G/G", "G/G", "G/G", "G/G", "G/G")

covariate<-c(1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1,
1, 1, 1, 0, 0, 0, 0, 0, 0)


response.q<-c(-0.3566, 0.4999, 0.9204, 0.8139, 1.8662, -0.4734, -0.7104,
-1.5904, -0.4367, 1.4043, 0.2683, 0.2324, 0.0217, 0.527, -2.2662,
0.1757, 0.378, 0.5139, -0.5167, -0.6913, 0.3385, -0.1095, -1.293,
1.7752, -1.4137, -0.502, -0.2889, -0.9809, -1.2051, -0.4104,
0.1964, -0.5435, 1.7636, 0.6596, 0.767, -0.4716, 1.1389, 0.8475,
-0.0428, 0.5691, 1.4069, -0.7324, 1.8495, -1.4328, 0.8782, -0.2168,
-0.006, 0.0517, 0.5135, 0.7965)

data.orig<-data.frame(snp1,snp2,covariate,response.q)

data<-setupData(data.orig,snps.name=c("snp1","snp2"),sep="/")

#before the execution of bayhapReg, execute bayhapFreq to have labels for each
#haplotype in the genotypes sample.

res.freq<-bayhapFreq(data=data,na.snp.action="keep",col.snps=1:2,sep="/",
                     total.iter.haplo=10000)

#res.freq, do not execute this!

# Haplotypes snp1 snp2    Freq Std.error     ICL     ICU
#    haplo.1    T    G 0.02473   0.01282 0.00297 0.05328
#    haplo.2    C    A 0.03602   0.01437 0.00624 0.06949
#    haplo.3    T    A 0.00000   0.00001 0.00000 0.00582
#    haplo.4    C    G 0.93924   0.02010 0.89985 0.98457

#From this object of class "freq" haplotype labels can be extracted for hap.prior,
#and then the prior values of the coefficients for the haplotypes chosen by the user
#and its standard deviation must be introduced.


#Put the word "haplotypes" in the formula. Model with interaction and with prior
#information.


res.q<-bayhapReg(formula=response.q~haplotypes+haplotypes*covariate,data=data,
                 family="gaussian",t.model="additive",na.snp.action="keep",
                 freqmin=0.01,burn.in.haplo=5000,burn.in.pheno=5000,
                 total.iter=5000,devhaplo=0.1,dist=1,lag.number=10,
                 sign=0.05,file=TRUE,prior.val=haplo.prior(res=res.freq,
                 haplos.name=c("haplo.1","haplo.2"),coeff=c(1.1,0.8),
                 sd.coeff=c(0.9,0.5)),verbose=2)
```

BIC *Bayesian Information Criterion*

Description

This function computes the Bayesian Information Criterion of a model.

Usage

```
BIC(res)
```

Arguments

res	An object of class reg returned by the function bayhapReg.

Details

The Bayesian information criterion (BIC) is a criterion for model selection among a class of parametric models with different numbers of parameters. BIC value is computed through the formula -2 log(L)+klog(n) where L is the maximized value of the likelihood function for the estimated model, k is the number of terms of the markov chain, i.e. the number of free parameters to be estimated (if the estimated model is a linear regression, k is the number of regressors, including the constant) and n is the sample size. If several models are runned, you can compare them by using the BIC criterion. The lower the BIC value, the better the model fit.

Value

The value returned is the BIC value.

Author(s)

Raquel Iniesta <riniesta@pssjd.org>

conv.test *Heidelberger and Welch Convergence Diagnostics*

Description

Computes the Heidleberger and Welch convergence diagnostics for the parameters in an MCMC sequence.

Usage

```
conv.test(x, alpha = 0.05, error = 1e-05, keep.rares.conv = FALSE)
```

Arguments

x	An object of class reg or class freq.
alpha	Alpha level for the confidence in the sample mean of the retained iterations.
error	Accuracy of the posterior estimates for the parameters.
keep.rares.conv	Logical. TRUE or FALSE indicating whether the diagnostic of convergence must be carried out also for the rares category.

Details

Take care when setting keep.rares.conv as TRUE. The chain for this parameter tends to be unstable and could lead to an error.

Value

A matrix whose columns and rows are the Heidleberger and Welch convergence diagnostics (i.e. stationarity test, number of iterations to keep and to drop, Cramer-von-Mises statistic, halfwidth test, mean, and halfwidth) and the monitored parameters, respectively.

Author(s)

Original version by Brian J. Smith, Nicky Best, Kate Cowles in Boa Package. Adapted version by Raquel Iniesta <riniesta@pssjd.org>

References

Heidelberger, P. and Welch, P. (1983). Simulation run length control in the presence of an initial transient. Operations Research, 31, 1109-44.

---

correl *Correlation Matrix*

Description

Computes and displays the correlation matrix for the parameters in the MCMC sequence.

Usage

```
correl(x, keep.rares.correl = FALSE)
```

Arguments

x	An object of class reg or class freq.
keep.rares.correl	Logical. TRUE or FALSE indicating whether the correlation must be carried out also for the rares category.

Value

The correlation matrix for the parameters in the MCMC sequence.

Author(s)

Original version by Brian J. Smith. Adapted version by Raquel Iniesta <riniesta@pssjd.org>

---

haplo.prior *Prior values for model parameters*

Description

Function only to be called inside bayhapReg

Usage

```
haplo.prior(res.freq=NULL,haplos.name=NULL,coeff=NULL,sd.coeff=NULL,
OR=NULL,CI.OR=NULL,sign.OR=NULL)
```

Arguments

res.freq	An object of class freq returned by bayhapFreq
haplos.name	Vector containing the name of the haplotypes for which prior values of model coefficient and standard deviation are given by the user.
coeff	Prior values for each model coefficient of the chosen haplotypes. These values must be specified for each haplotype respectively in the same order stated in haplos.name. An analysis without prior information or with only some haplotypes with prior information is allowed.
sd.coeff	Prior values for each sd of each model coefficient of the chosen haplotypes. These values must be specified for each haplotype respectively in the same order stated in haplos.name. An analysis without prior information or with only some haplotypes with prior information is allowed.
OR	If the family model is the binomial, this must be a vector of prior OR's for each haplotype name. The order must be the same as in haplos.name
CI.OR	If the family model is the binomial and you have prior information for OR values, this must be a matrix with two columns and so many rows as the number of haplotypes with prior information. Each row represents a haplotype. For each haplotype the first column is the lower limit of the confidence interval, and the second column is the upper limit.
sign.OR	In case of a binomial model with OR and CI.OR informed, this must be a vector containning each significance used in the computation of each CI.OR.

Details

In order to get the internal haplotype labels assigned by the package, first run bayhapFreq. This object is also needed in haplo.prior. It is not necessary to put prior values for all haplotypes in the sample, only choose the ones for which you have prior information. For them, the model coefficient and the standard deviaton for the coefficient must be given. If the family model is the binomial, there are two possible options. You can put prior information for model coefficients (mean and standard deviation) or you can put prior information for the OR, for the confidence interval and significance of each interval, instead of inform about the coefficients.

For a complet example see the help of bayhapReg.

Value

Internal values only used as a parameter information by the function bayhapReg.

Author(s)

Raquel Iniesta <riniesta@pssjd.org>

---

plotACF *Plot autocorrelation function*

Description

Creates a single plot of the lag autocorrelations for a specified parameter.

Usage

```
plotACF(x, all = TRUE, name.var = " ", keep.rares.acf = FALSE)
```

Arguments

x	An object of class reg or freq.
all	Logical.The default value TRUE implies to plot autocorrelation function for all the parameters. If FALSE, the vector of names of the desired parameters to plot must be specified in name.var.
name.var	If all=FALSE, this must be a vector containing the names of the chosen parameters to plot their acf.
keep.rares.acf	Logical. Indicates if acf for estimates of rares category must be shown.

Value

Graphs of autocorelation function.

Author(s)

Original version by Brian J. Smith in Boa package.

Adapted version by Raquel Iniesta <riniesta@pssjd.org>

plotDensity *Plot Density Functions*

Description

Estimates and displays the density function(s) for the specified parameter(s).

Usage

```
plotDensity(haplo.object, all = TRUE, name.var = " ",
            keep.rares.density = FALSE)
```

Arguments

haplo.object	An object of class reg or freq.
all	Logical.The default value TRUE implies to plot the density function for all the parameters in the haplo.object. If FALSE, the vector of names of the desired parameters to plot must be specified in name.var.
name.var	If all=FALSE, this must be a vector containing the names of the chosen parameters to plot their density.
keep.rares.density	Logical. Indicates if the density for rare category must be plotted.

Value

Graphs of density function.

Author(s)

Original version by Brian J. Smith.

Adapted version by Raquel Iniesta <riniesta@pssjd.org>

plotFreq *Plot acf, running mean, density functions and trace histories*

Description

Displays plots of the autocorrelation function, the running mean, the density functions and the trace histories for the parameter(s) in the MCMC sequence. This function is suitable to plot objects of class freq.

Usage

```
plotFreq(res, acf = NULL, d = NULL, rmean = NULL, tr = NULL)
```

Arguments

res	An object of class freq.
acf,d,rmean,tr	If all these arguments are set to the default NULL, plotFreq will plot all the graphs (auctocorrelation, density, running mean and trace). To plot only one, two or three graphs, set one, two or three of these arguments as TRUE.

Details

To plot only one type of graph, you can also use the functions plotACF, plotRmean, plotTrace and plotDensity.

Value

Plots of autocorrelation functions, density functions, running mean and trace histories.

Author(s)

Original version by Brian J. Smith. Adapted version by Raquel Iniesta. <riniesta@pssjd.org>

See Also

plotACF, plotTrace, plotRmean, plotDensity

---

plotReg *Plot acf, running mean, density functions and trace histories*

Description

Displays plots of the autocorrelation function, the running mean, the density functions and the trace histories for the parameter(s) in the MCMC sequence. This function is suitable to plot objects of class reg.

Usage

```
plotReg(res, keep.rares.plot = FALSE, acf = NULL, d = NULL,
        rmean = NULL, tr = NULL)
```

Arguments

res	An object of class reg.
keep.rares.plot	Logical. Indicates if plots for estimates of rare category must be shown.
acf,d,rmean,tr	If all these arguments are set to the default value NULL, plotReg is going to plot all the graphs (auctocorrelation, density, running mean and trace). To plot only one, two or three graphs, set one, two or three of these arguments as TRUE.

Details

To plot only one type of graph, you can also use the functions `plotACF`, `plotRmean`, `plotTrace` and `plotDensity`.

Value

Plots of autocorrelation functions, density functions, running mean and trace histories.

Author(s)

Original version by Brian J. Smith.

Adapted version by Raquel Iniesta. <riniesta@sjd-ssm.com>

See Also

plotACF, plotTrace, plotRmean, plotDensity

---

plotRmean *Plot Running Mean of the Parameter Estimation*

Description

Computes and displays the running mean(s) for the specified parameter(s).

Usage

```
plotRmean(haplo.object, all = TRUE, name.var = " ",
          keep.rares.rmean = FALSE)
```

Arguments

`haplo.object`	An object of class `reg` or `freq`.
`all`	Logical.The default value TRUE implies to plot the running mean for all the parameters. If FALSE, the vector of names of the desired parameters to plot must be specified in name.var.
`name.var`	If all=FALSE, this must be a vector containing the names of the chosen parameters to plot their running mean.
`keep.rares.rmean`	Logical. Indicates if the running mean for rare category must be shown.

Value

Graphs of running mean(s).

Author(s)

Original version by Brian J. Smith in package Boa.

Adapted version by Raquel Iniesta <riniesta@pssjd.org>.

plotTrace *Plot Trace Histories*

Description

Displays the trace histories for the specified parameter(s).

Usage

```
plotTrace(haplo.object, all = TRUE, name.var = " ",
          keep.rares.trace = FALSE)
```

Arguments

haplo.object	An object of class reg or freq.
all	Logical. The default value TRUE implies to plot the trace for all the parameters. If FALSE, the vector of names of the desired parameters to plot must be specified in name.var.
name.var	If all=FALSE, specify here a vector containing the names of the chosen parameters to plot their trace.
keep.rares.trace	Logical. Indicates if the trace for rare category must be shown.

Value

Graphs of trace histories.

Author(s)

Original version by Brian J. Smith in package Boa.

Adapted version by Raquel Iniesta <riniesta@pssjd.org>

setupData *Check and prepares original data to be analyzed*

Description

This function checks if the data set is suitable for the analysis (type and format) and creates a new data frame including some internal variables.

Usage

```
setupData(data, col.snps = NULL, snps.name = NULL, sep)
```

Arguments

data	A data.frame in which each row represents an individual and columns are the variables ocurring in the formula (response, SNPs information and covariates, see bayhapReg documentation for more details). All columns must be non factors. Each SNP have to be a character string holding both alleles with the separator specified by the argument sep. Data must contain one column for each SNP. No variable in the original data.frame should be called haplotypes by the user.
col.snps	A numerical vector indicating the position of each SNP in data. This is an optional parameter, instead of this the name of each SNP can be informed in snps.name.
snps.name	A vector containing the names of each SNP in the data. This is an optional parameter, instead of this the column position of each SNP can be informed in col.snps.
sep	Character separator to divide alleles.

Details

This function must be executed before running bayhapReg. The resulting data.frame must be the object to pass to bayhapReg as a data parameter. Original data must not contain any variable named haplotypes because this is an specific word for this package.

Value

A data.frame ready to be inserted in bayhapReg.

Author(s)

Raquel Iniesta <riniesta@pssjd.org>

See Also

bayhapReg

Index