Package ‘MISA’

November 3, 2010

Type Package

Title Bayesian Model Search and Multilevel Inference for SNP Association Studies

Author Melanie Wilson `<maw27@stat.duke.edu>`

Maintainer Gary Lipton `<gl37@stat.duke.edu>`

Description The functions in this package focus on intermediate throughput case-control association studies, where the outcome of interest is often a binary disease state and where the genetic markers have been chosen to capture variation in a set of related genes, such as those involved in a specific biochemical pathway. Given this data, we are interested in addressing two questions: “To what extent does the data support an overall association between the pathway and outcome of interest?” and “Which markers or genes are most likely to be driving this association?” To address both of these questions,this package performs a Bayesian model search technique that utilizes Evolutionary Monte Carlo and searches over models including main effects of all genetic markers and marker-specific genetic effects in a computationally efficient manner. The package incorporates functions that perform a marginal screen on the genetic markers, summarize the output of the model search algorithm, including image plots of the models with the highest posterior probability, marginal summaries of SNP and gene inclusion probabilities and Bayes Factors, and global summaries of the posterior probability and Bayes Factor giving evidence of an association in the set of SNPs of interest.

Version 2.10.0-6.7.2

License Unlimited

Depends coda, BAS (>= 0.92)

Suggests multicore, tcltk

R topics documented:

`MISA-package` *Multilevel Inference for SNP Association Studies*

Description

The functions in this package focus on intermediate throughput case-control association studies, where the outcome of interest is often a binary disease state and where the genetic markers have been chosen to capture variation in a set of related genes, such as those involved in a specific biochemical pathway. Given this data, we are interested in addressing two questions: ”To what extent does the data support an overall association between the pathway and outcome of interest?” and ”Which markers or genes are most likely to be driving this association?” To address both of these questions, this package performs a Bayesian model search technique that utilizes Evolutionary Monte Carlo and searches over models including main effects of all genetic markers and marker-specific genetic effects in a computationally efficient manner. The package incorporates functions that perform a marginal screen on the genetic markers, summarize the output of the model search algorithm, including image plots of the models with the highest posterior probability, marginal summaries of SNP and gene inclusion probabilities and Bayes Factors, and global summaries of the posterior probability and Bayes Factor giving evidence of an association in the set of SNPs of interest.

Details

Package:	MISA
Version:	2.6.5
Date:	2010-07-08
Depends:	R (>= 2.10.0), BAS (>= 0.92)
License:	GPL-2
URL:	http://www.stat.duke.edu/gbye/MISA.html

Author(s)

Melanie Wilson,
Maintainer: Gary Lipton <gl37@stat.duke.edu>

References

Geyer C (1991). *Markov chain Monte Carlo maximum likelihood.* Computing Science and Statistics:156.

Holland J (1975). *Adaptation in Natural and Artificial Systems.* University of Michigan Press.

Liang F, Wong W (2000). *Evolutionary Monte Carol:Applications to Cp Model Sampling and Change Point Problem.* Statistica Sinica 10:317.

See Also

`bas`

`bf4assoc` *Bayes Factors for 3 Association Models: Marginal SNP Screen*

Description

This function performs a marginal screen on the SNPs of interest by using Laplace approximations to estimate the marginal Bayes Factors (BFs) of each SNP. In particular, we estimated the marginal likelihood of each of the three genetic models of association (log–additive, dominant and recessive) and under the null model (model of no genetic association). The BF for a model of association is defined as the ratio of the marginal likelihood of that model of association to the marginal likelihood of the null model. We can then use a decision rule such as including only the SNPs with a maximum marginal Bayes factor for each genetic model greater than 1.

Usage

```
bf4assoc(D, X = NULL, XS, Ns, Nx, snpsd, Prior, MinCount, MaxIt = 100,
         RelTol = 1e-4, scoring = 1)
```

Arguments

`D`	response vector coded (0=control, 1=case) of length N.
`X`	numeric matrix of confounder/design variables of dimension N by Nx is NULL if Nx==0.
`XS`	numeric matrix of SNP variables of dimension N by Ns coded (0 = common homozygote, 1 = heterozygote, 2 = rare homozygote, 3 = missing).
`Ns`	number of SNPs in the input data set.
`Nx`	number of design/confounder variables included in all models.
`snpsd`	standard deviation of mean zero prior on the genetic effect parameter when Prior==0 and scale when Prior==1.
`Prior`	set to 0 to chose Normal prior and 1 to chose Cauchy prior.
`MinCount`	count below which a genotype is treated as absent for a given SNP. Effect is to reduce the number of unique genetic models that are discernable for that SNP.
`MaxIt`	maximum number of function evaluations per optimization.
`RelTol`	relative tolerance.
`scoring`	optimization algorithm; must be either scoring (scoring = 1) or Newton-like (scoring = 0).

Details

Use Laplace Approx to calculate Bayes factors in favor of 3 genetic models of association. Features:(1) Laplace approximation based estimates. (2) Output log-ORs + SE(logOR)s.(3) Option for Cauchy prior.

Value

This function outputs a matrix of the following values:

`SNP`	SNP ID number.
`N.geno`	number of genotypes with counts exceeding MinCount among both cases and controls. If 1, the snp is treated as if it were monomorphic – no models are fit; if 2, only the log-additive model is fit.
`bfAtoN`	marginal likelihood ratio: Pr(Data|logadditive model)/Pr(Data|null model) ie BF for log-additive model.
`bfDtoN`	marginal likelihood ratio: Pr(Data|dominant model)/Pr(Data|null model) or BF for dominant model.
`bfRtoN`	marginal likelihood ratio: Pr(Data|recessive model)/Pr(Data|null model) or BF for recessive model.
`PrAgvnAssoc`	posterior probability of a log-additive genetic model given an association.
`PrDgvnAssoc`	posterior probability of a dominant genetic model given an association.
`PrRgvnAssoc`	posterior probability of a recessive genetic model given an association.
`logOR.LogAdd`	modal estimate of log odds ratio under the log-additive model.
`logOR.Dom`	modal estimate of log odds ratio under the dominant model.
`logOR.Rec`	modal estimate of log odds ratio under the recessive model.
`SE.lOR.LogAdd`	estimate of the standard error for the modal estimate of the log odds ratio for the SNP variable under the log-additive model.
`SE.lOR.Dom`	estimate of the standard error for the modal estimate of the log odds ratio for the SNP variable under the dominant model.
`SE.lOR.Rec`	estimate of the standard error for the modal estimate of the log odds ratio for the SNP variable under the recessive model.

Author(s)

Ed Iversen <iversen@stat.duke.edu>

Examples

```
## Load the data
data(dna.snp.full)
## Find the number of snps in the data set
p <- (dim(dna.snp.full)[2] - 2)

## Calculate the Marginal BF's for the SNPs
marg.bf <- bf4assoc(D=dna.snp.full$case,
                    X=as.matrix(dna.snp.full$age),
                    XS=as.matrix(dna.snp.full[,-c(1,2)]),
                    Ns=p, Nx=1, snpsd=0.25, Prior=0, MinCount=1.9,
                    MaxIt=1000, RelTol=1e-7)
## Calculate the Maximum BF for each SNP (LA, Dom, Rec)
```

```
max.bf <- apply(marg.bf[,c(3:5)], 1, max)

## Screen the data based on max.bf > 1
dna.snp <- dna.snp.full[, c(TRUE, TRUE, max.bf > 1)]
```

`combine.EMC` *Calculates Global and Marginal Summaries*

Description

This function calculates the global and marginal posterior probabilities and Bayes Factors that give the evidence of there being an association in the overall set of SNPs of interest, the individual genes of interest and the individual SNPs of interest.

Usage

```
combine.EMC(emc.list)
```

Arguments

`emc.list`	A list of output structures from `Gene.EMC`

Details

This function consolidates multiple outputs from `Gene.EMC` into a single structure.

Value

A structure of the form generated by `Gene.EMC` combining the results in emc.list.

Author(s)

Gary Lipton <gl37@stat.duke.edu>

Examples

```
data(emc.out.1)
data(emc.out.2)
combo.out <- combine.EMC(list(emc.out.1, emc.out.2))
```

`converge.EMC` *Convergence Diagnostic Plots for Genetic EMC*

Description

Takes the output of two independent runs of the Genetic EMC sampling algorithm and creates plots to assess the convergence of the algorithm.

Usage

```
converge.EMC(emc.out, plot.type, bandwidth = 1000, a = 1, b = NULL,...)
```

Arguments

`emc.out`	output from `Gene.EMC`
`plot.type`	indicates the type of plot; options are ”iter”, ”gelman.rubin”,”bayes.factor”, ”snp.inc”, or ”all”.
`bandwidth`	If the ”bayes.factor” plot.type is chosen, user must indicate the bandwidth of iterations over which to calculate the global Bayes Factor over.
`a`	If ”bayes.factor” or ”snp.inc” plot.type is chosen, and the fitness function used in the Genetic EMC algorithm is ”AIC.BB” the user must specify the value of a for the beta hyper-parameter.
`b`	If ”bayes.factor” or ”snp.inc” plot.type is chosen, and the fitness function used in the Genetic EMC algorithm is ”AIC.BB” the user must specify the value of b for the beta hyper-parameter.
`...`	general parameters for plotting functions.

Details

The four plot types are described as follows:

iter: Plot of the cost values over each iteration for each of the independent runs. These trace plots help to examine if a balanced has been reached between exploring the model space and convergence rates. Make sure that the cost values are not sticking too much in one area and are moving around freely to explore the space properly. If the trace plots do show that the algorithm is tending to get stuck in on area this may be a sign that you need to increase the max temperature or increase the number of parallel chains so that adjacent chains can communicate better.

gelman.rubin: Plot of the gelman rubin convergance diagnostic (see Gleman, Rubin (1992)) of the cost values of the two independent chains.

bayes.factor: Plot of the global Bayes factor computed across iterations for each independent chain. Since our global Bayes factor is a lower bound for the global Bayes factor computed if we were able to enumerate all models the Bayes factor will increase for every new unique model that we find at any of the iterations. Therefore, we are interested in seeing if the Bayes factor begins to converge after a given number iterations and is no longer making large jumps.

snp.inc: Plot of the Marginal Bayes factors for one independent run vs. another independent run. This plot enables us to determine if the values of the marginal SNP inclusion probabilities are consistent across two independent runs of the algorithm. If the plot does not following the line y = x then this is an indication that the algorithm as not yet converged and is still exploring the space.

Value

This function plots four convergence diagnostics for the Genetic EMC algorithm.

Author(s)

Melanie Wilson <maw27@stat.duke.edu>

References

Gelman A, Rubin D (1992). *Inference from iterative simulation using multiple sequences.* Statistical Science 7:457.

Examples

```
##Load the emc.out files for two independent runs of the dna.snp data
data(emc.out.1)
p <- dim(emc.out.1$which)[2] - 1
data(emc.out.2)
emc.out <- list(emc.out.1, emc.out.2)

##Look at all of the convergence plots for the dna.snp data
converge.EMC(emc.out,plot.type="all", bandwidth=100 ,b=p, a=1)
```

`crossover` *EMC: Crossover Step*

Description

This function takes the current state of the population in the Genetic EMC algorithm and performs the crossover step.

Usage

```
crossover(pop, pop.fit, cross.a, force, data, fitness, t,
          impute = impute, b = NULL, a = 1, rec.mdl,
          cores = cores)
```

Arguments

`pop`	matrix specifying the current status of each chain (or model) of the population.
`pop.fit`	vector of fitness values for each of the models specified in the current population.
`cross.a`	current number of accepted crossover steps.
`force`	character vector specify the variables to force in the models
`data`	data frame of the same form in `Gene.EMC`
`fitness`	character string specifying the fitness function to use in the algorithm.
`t`	temperature vector specifying the temperature value for each chain in the population.
`impute`	number of imputed data sets.

`b`	If the fitness function is "AIC.BB", the user must specify the value for the beta hyper-parameter b.
`a`	If the fitness function is "AIC.BB", the user must specify the value for the beta hyper-parameter a.
`rec.mdl`	indicator vector that indicates which SNPs should not have a recessive parameter since the power is too weak.
`cores`	number of cores to use; i.e. the maximum number of processes to spawn.

Details

In the crossover step, one of the top current models from the population is chosen to mate with another random model and two new models are formed by some composition of the two parental models.

Value

This function outputs a list of the following values:

`pop`	current status of the population.
`pop.fit`	vector of fitness values for each of the models specified in the current population.
`cross.a`	current number of accepted crossover steps.

Author(s)

Melanie Wilson <maw27@stat.duke.edu>

References

Geyer C (1991). *Markov chain Monte Carlo maximum likelihood.* Computing Science and Statistics:156.

Holland J (1975). *Adaptation in Natural and Artificial Systems.* University of Michigan Press.

Liang F, Wong W (2000). *Evolutionary Monte Carlo:Applications to Cp Model Sampling and Change Point Problem.* Statistica Sinica 10:317.

`dna.snp` *Simulated Example SNP data set.*

Description

Simulated data set of 26 SNP variables in the DNA repair pathway. The first column represents the disease status of the individual, the second column the forced variable of age and in the remaining columns we have the log-additive parameterizations of the SNP variable. This is the subset data set of dna.snp.full once the marginal Bayes factor screen was performed.

Usage

```
data(dna.snp)
```

Format

A data frame with 1197 observations on the following 19 variables (case, age, snp.la).

Details

The simulated samples comprised a binary outcome and genetic data on 399 cases and 798 controls where the genetic data was simulated at the same 60 tag SNPs as genotyped in a particular study of interest on the DNA repair pathway. We simulated the genotypes in two stages. First, for each of the 6 genes represented in the data set, we phased the NCOCS control SNP genotype data and estimated recombination rates using PHASE (Stephens et al., 2001). Second, given a model of association, we generated case-control data at these tags using HAPGEN (Marchini and Su, 2006). For this simulation, we assumed that a randomly chosen subset of 2 genes were associated and that, within the associated genes, a single, randomly chosen tag was the source of the association. One of the associated tags SNPs were accorded an odds ratio (OR) of 1.75 and an assumed genetic log-additive genetic parametrization and one an OR of 1.25 and a dominate genetic parametrization. To facilitate the mixing of the Genetic EMC algorithm and ease computation time we screened the SNPs in each full simulation to the final subset of SNPs seen in this data set with a marginal Bayes factor estimated to be 1.0 or above by a procedure described in `bf4assoc`. Information on the SNPs in this simulation can be found in the data set `sim.info`.

References

Stephens M, Smith N, Donnelly P (2001). A New Statistical Method for Haplotype Reconstruction from Population Data. The American Journal of Human Genetics 68:978-989.

Marchini J, Su Z (2006). HAPGEN, a C++ program for simulating case and control SNP haplotypes.

Examples

```
data(dna.snp)
```

`dna.snp.full` *Simulated Example SNP data set.*

Description

Simulated data set of 60 SNP variables in the DNA repair pathway. The first column represents the disease status of the individual, the second column the forced variable of age, and the remaining columns the log-additive parametrizations of each SNP variable. This is the full simulated data set before the marginal screen was performed.

Usage

```
data(dna.snp)
```

Format

A data frame with 1197 observations on the following 60 variables (case, age, snp.la).

Details

The simulated samples comprised a binary outcome and genetic data on 399 cases and 798 controls where the genetic data was simulated at the same 60 tag SNPs as genotyped in a particular study of interest on the DNA repair pathway. We simulated the genotypes in two stages. First, for each of the 6 genes represented in the data set, we phased the NCOCS control SNP genotype data and estimated recombination rates using PHASE (Stephens et al., 2001). Second, given a model of association, we generated case-control data at these tags using HAPGEN (Marchini and Su, 2006). For this simulation, we assumed that a randomly chosen subset of 2 genes were associated and that, within the associated genes, a single, randomly chosen tag was the source of the association. One of the associated tags SNPs were accorded an odds ratio (OR) of 1.75 and an assumed genetic log-additive genetic parametrization and one an OR of 1.25 and a dominate genetic parametrization. Information on the SNPs in the simulation can found in the data set `sim.info`.

References

Stephens M, Smith N, Donnelly P (2001). A New Statistical Method for Haplotype Reconstruction from Population Data. The American Journal of Human Genetics 68:978-989.

Marchini J, Su Z (2006). HAPGEN, a C++ program for simulating case and control SNP haplo- types.

Examples

```
data(dna.snp.full)
```

emc.out.1 *Example Output I from the Genetic EMC algorithm*

Description

Output from the first independent run of the Genetic EMC algorithm for the dna.snp data set. The algorithm was run for approximately 500,000 iterations. The assumed fitness function was ”AIC.BB” where we assumed a Beta-Binomial prior on the size of the models sampled with hyper-parameters on the beta distribution, a=1 and b=p (p=82).

Usage

```
data(emc.out.1)
```

Format

Output from the function `Gene.EMC` that is a list of the following values:

which: Matrix where each row corresponds to a model specification vector for the unique models visited and the value of the fitness function for the model.

data: Data frame of the same form as the imputed data frame.

iter.aic: The value of the fitness function for each of the models visited at each of the iterations.

iter: The total number of iterations run.

iter.unique: Vector indicating the number of the iteration in which each of the unique models was found.

force: Character vector indicating the names of the forced variables.

fitness: Character string indicating the fitness function used for the algorithm.

Examples

```
data(emc.out.1)
```

---

emc.out.2 *Example Output II from the Genetic EMC algorithm*

Description

Output from the second independent run of the Genetic EMC algorithm for the dna.snp data set. The algorithm was run for approximately 500,000 iterations. The assumed fitness function was "AIC.BB" where we assumed a Beta-Binomial prior on the size of the models sampled with hyper-parameters on the beta distribution, a=1 and b=p (p=82).

Usage

```
data(emc.out.2)
```

Format

Output from the function `Gene.EMC` that is a list of the following values:

which: Matrix where each row corresponds to a model specification vector for the unique models visited and the value of the fitness function for the model.

data: Data frame of the same form as the imputed data frame.

iter.aic: The value of the fitness function for each of the models visited at each of the iterations.

iter: The total number of iterations run.

iter.unique: Vector indicating the number of the iteration in which each of the unique models was found.

force: Character vector indicating the names of the forced variables.

fitness: Character string indicating the fitness function used for the algorithm.

Examples

```
data(emc.out.2)
```

`exchange` *EMC: Exchange Step*

Description

This function takes the current state of the population in the Genetic EMC algorithm and performs the exchange step between each neighboring chain.

Usage

```
exchange(pop, pop.fit, exc.a, temp)
```

Arguments

`pop`	matrix specifying the current status of each chain (or model) of the population.
`pop.fit`	vector of fitness values for each of the models specified in the current population.
`exc.a`	current number of accepted crossover steps.
`temp`	temperature vector specifying the temperature value for each chain in the population.

Details

This step corresponds to the normal parallel tempering exchange step, where we allow the models, or current states of the chains to move up and down the temperature ladder.

Value

This function outputs a list of the following values:

`pop`	current status of the population.
`pop.fit`	vector of fitness values for each of the models specified in the current population.
`exc.a`	current number of accepted exchange steps.

Author(s)

Melanie Wilson <maw27@stat.duke.edu>

References

Geyer C (1991). *Markov chain Monte Carlo maximum likelihood.* Computing Science and Statistics:156.

Holland J (1975). *Adaptation in Natural and Artificial Systems.* University of Michigan Press.

Liang F, Wong W (2000). *Evolutionary Monte Carol:Applications to Cp Model Sampling and Change Point Problem.* Statistica Sinica 10:317.

`expand.data.snp` *Data: Genetic Parametrization*

Description

This function takes the data set with log-additive parameterization for each SNP and makes a data set with log-additive, dominant, and recessive parameterization.

Usage

```
expand.data.snp(data.snp, ind.info, force, subset)
```

Arguments

`data.snp`	(n x p) matrix of log-additive SNP genotypes for each individual where 0 is the hom. common, 1 het. and 2 hom. rare genotype.
`ind.info`	(n x (q+1)) matrix of information on each individual where q is the number of forced variables in the study. The first column must represent the case/control status followed by a column for each of the forced variables.
`force`	vector of variable names that you wish to force into the final model. These variables must be found in the ind.info matrix.
`subset`	vector of TRUE/FALSE indicating the individuals you want to include in the study

Value

This function outputs a data frame where the first column is the case/control variable, the next columns are the forced variables and the last columns are the SNP.la, SNP.dom and SNP.rec variables, respectively.

Author(s)

Melanie Wilson <maw27@stat.duke.edu>

`fit.EMC` *EMC: Fitness Function*

Description

This function takes one of the models in the population (or a current state of a chain) and calculates the fitness/cost value of the model.

Usage

```
fit.EMC(samp, force = NULL, data, fitness = "AIC", impute = impute, b =
        NULL, a = 1, rec.mdl)
```

Arguments

`samp`	vector specifying the current value of one of the chains of the population: or a model specification vector.
`force`	character vector specify the variables to force in the model.
`data`	data frame of the same form in `Gene.EMC`.
`fitness`	character string specifying the fitness function to use in the algorithm.
`impute`	number of imputed data sets.
`b`	If the fitness function is ”AIC.BB”, this specifies the hyper-parameter in the beta distribution on the model prior.
`a`	If the fitness function is ”AIC.BB”, this specifies the hyper-parameter in the beta distribution on the model prior.
`rec.mdl`	Indicator vector that indicates which SNPs should not have a recessive parameter since the power is too weak.

Details

The different options for the fitness function are ”AIC”, ”BIC”, or ”AIC.BB” where we add in a penalty to each of the models that corresponds to a beta-binomial prior on the model size with a and b chosen by the user.

Value

The function returns the cost value (-1/2*fitness value) of the model corresponding to the model specification vector given in samp.

Author(s)

Melanie Wilson <maw7@stat.duke.edu>

References

Geyer C (1991). *Markov chain Monte Carlo maximum likelihood.* Computing Science and Statistics:156.

Holland J (1975). *Adaptation in Natural and Artificial Systems.* University of Michigan Press.

Liang F, Wong W (2000). *Evolutionary Monte Carol:Applications to Cp Model Sampling and Change Point Problem.* Statistica Sinica 10:317.

`Gene.EMC`	*Bayesian Model Search Algorithm for Case-Control Genetic Association Studies*

Description

This function performs a sampling algorithm that is based on Evolutionary Monte Carlo (EMC) (Liang and Wong, 2000) which is a combination of parallel tempering (Geyer, 1991) and the genetic algorithm (Holland, 1975). The basic idea behind the genetic algorithm is taken from evolution in that individuals in a population compete and mate in order to produce increasingly stronger individuals. Here, the individuals correspond to models of SNP data and the population corresponds to a set of models of interest. The strength of each model is determined by the user specified fitness function and we are therefore interested in sampling the strongest models on basis of the given fitness function.

Usage

```
Gene.EMC(data, force = NULL, fitness = "AIC.BB", b = NULL, a = 1,
         impute = 1, snp.subset = NULL, start.snps = NULL,
         n.iter = 10000, N = 5, tmax = 1, tlone = 0, qm = 0.25,
         display.acc = FALSE, display.acc.ex = FALSE,
         status.out = NULL, chkpt.out = "chkpt.rda",
         save.iter = 10000, burnin = 1, cores = 1, log = TRUE,
         pb = FALSE)
```

Arguments

`data`	a data frame (or a list of data frames for multiple imputed data sets) where the first column corresponds to the response variable of interest (disease status), the next nF columns are the forced variables of interest, and the final p columns are the SNPs of interest with a log-additive parameterization:

[,1]	case	multiple	Response (0|1)
[,2]	covariate_1	numeric	Forced variable
	.	numeric	Forced variable
	.	numeric	Forced variable
[,p+1]	covariate_nF	numeric	Forced variable
[,nF+2]	SNP_1	numeric	Copy number (0|1|2)
	.	numeric	Copy number (0|1|2)
	.	numeric	Copy number (0|1|2)
[,nF+p+2]	SNP_p	numeric	Copy number (0|1|2)

”Case” may be presented as a numeric, factor, or logical variable. All values will be converted to numeric. For factors, the log file will report the correspondence between factor levels and numeric values.

`force`	a character vector of variable names to be forced into the sampled models. These variable names should correspond to the column headers for these variables in data.
`fitness`	a character string that specifies the fitness function to be used. Options are ”AIC”,”BIC”,”AIC.BB”.
`b`	the Beta hyper-parameter b, when the fitness function is ”AIC.BB”.
`a`	the Beta hyper-parameter a, when the fitness function is ”AIC.BB”.
`impute`	the number of data sets on which to run the algorithm when data is a list of data frames for multiple imputed data sets.

`snp.subset`	a logical vector that indicates which SNPs (in the same order as the data frame) to run the algorithm on. This variable can be useful for searching through only a number of SNPs that have passed a set screen.
`start.snps`	a logical vector of the SNPs that will be in the initial model as a log-additive parameterization. If NULL the algorithm initializes with 5 random SNPs with the log-additive parametrization.
`n.iter`	the number of iterations to run the algorithm.
`N`	the number of parallel chains.
`tmax`	the maximum temperature value of the chains. Default is to set tmax=1 so that there is a constant temperature ladder.
`tlone`	the number of chains that the user wants to run at a temperature value less than zero.
`qm`	the probability of the mutation update.
`display.acc`	logical flag indicating whether to output the acceptance rates of the mutation and crossover steps at each iteration.
`display.acc.ex`	logical flag indicating whether to output the acceptance rates of the exchanges between each parallel chain for each iteration. Helps identify whether the chains are too far apart and the temperature scheme needs to be changed.
`status.out`	character string giving the pathname of the file to write the status of the algorithm and the acceptance rates to, instead of stdout.
`chkpt.out`	character string giving the pathname of the checkpoint file to save the output of the algorithm to.
`save.iter`	the number of iterations between each checkpoint. A checkpoint file is written every save.iter iterations.
`burnin`	integer indicating the length of the burnin.
`cores`	the number of cores to use; i.e. the maximum number of processes to spawn.
`log`	logical flag indicating whether to write warnings and errors to a time-stamped log file.
`pb`	display X11 progress bar.

Details

The algorithm is run for a chosen number of iterations where we update the population via the mutation, crossover, and exchange steps of the genetic algorithm. For each iteration the algorithm runs the mutation operator with probability qm and the crossover operator with probability (1-qm) and then performs the exchange step with all adjacent chains. The user must specify a the population size, N, the maximum and minimum temperature for the ladder T. (We make the assumption that the jumps in the temperature ladder have the form tj-ti = exp(tj/ti) where ti = 1 for some value i in the ladder). These parameters are chosen so that the algorithm converges at a fast rate and can normally be determined in test runs of the algorithm by examining overall acceptance rates of the mutation and crossover rates and acceptance rates of exchanges between adjacent chains. Low exchange rates between chains indicates that the temperature values are too far apart for adjacent chains and either the max temp. should be decreased or the number of parallel chains should be increased. Also, convergence of the algorithm can be determined by running two independent runs of the algorithm with different starting values and examining the convergence plots produced in `emc.converge`.

Value

This function outputs a list of the following values to the file write.out if this file is specified for every save.iter number of iterations:

`which`	Matrix where each row corresponds to a model specification vector for the unique models visited and the value of the fitness function for the model.
`data`	Data frame used to run the algorithm of the same form as the imputed data frame.
`iter.aic`	The value of the cost function for each of the models visited at each of the iterations.
`iter`	The total number of iterations run.
`iter.unique`	Vector indicating the number of the iteration in which each of the unique models were found.
`force`	Character vector indicating the names of the forced variables.
`fitness`	Character string indicating the fitness function used for the algorithm.

Author(s)

Melanie Wilson <maw27@stat.duke.edu>

References

Geyer C (1991). *Markov chain Monte Carlo maximum likelihood.* Computing Science and Statistics:156.

Holland J (1975). *Adaptation in Natural and Artificial Systems.* University of Michigan Press.

Liang F, Wong W (2000). *Evolutionary Monte Carol:Applications to Cp Model Sampling and Change Point Problem.* Statistica Sinica 10:317.

Examples

```
## Load the data
data(dna.snp)
## Find the number of snps in the data set
p <- (dim(dna.snp)[2] - 2)

## Set the algorithm to start with SNPS 3,7, and 21.
start.snp <- rep(FALSE, p)
start.snp[c(3, 7)] <- TRUE

## Run algorithm for 100 iterations and save output to emc.out.
emc.out <- Gene.EMC(data=dna.snp, force=c("age"), fitness="AIC.BB",
                    b=p, a=1, start.snps=start.snp, n.iter=100, N=5,
                    tmax=5, tlone=1, qm=.25, display.acc=TRUE,
                    display.acc.ex=TRUE, cores=1)
```

model.inc *Image Plots for top SNP and Gene Inclusions*

Description

This function allows the user to create image plots of the top SNPs and top Genes included in the top models. For the SNP inclusion plots, the color of the inclusion block signifies genetic mode of inheritance for the specified SNP in each of the modes with: Purple = Log-Add., Red= Dom., Blue = Rec. SNPs and Genes are ordered based on marginal SNP inclusion probabilities which are plotted on the right axis. The width of the inclusion blocks are proportional to the posterior model probability that the SNP or Gene is included in.

Usage

```
model.inc(prob.out, num.models = 100, num.snps = 20, num.genes = 20,
          inc.typ = "s", hide.name = FALSE, ...)
```

Arguments

`prob.out`	output list from `post.prob`.
`num.models`	the number of the top models to place on the x-axis.
`num.snps`	If inc.type="s", the number of the top SNPs to place on the y-axis.
`num.genes`	If inc.type="g",the number of the top genes to place on the y-axis.
`inc.typ`	specifies if we want to plot the SNP inclusion ("s") or gene inclusion ("g")
`hide.name`	logical indicator determining if we should hide the specific gene and SNP names.
`...`	General parameters for plotting functions

Author(s)

Melanie Wilson <maw27@stat.duke.edu>

Examples

```
## Load the emc.out files for one run of the dna.snp data (100,000
## iterations)
data(emc.out.1)
data(sim.info)
p <- dim(emc.out.1$which)[2]-1

## Calculate posterior summaries for the output of the Genetic EMC algorithm
post.prob.out <- post.prob(emc.out.1, sim.info, b=p)

## Plot the SNP Inclusions in the top 100 models for the top 20 SNPs
model.inc(post.prob.out, num.models=100, num.snps=20, inc.typ="s")

## Plot the Gene Inclusions in the top 100 models for the top 20 Genes
model.inc(post.prob.out,num.models=100,num.genes=6,inc.typ="g")
```

`mutation` *EMC: Mutation Step*

Description

This function takes a current state of one of the chains of the population in the Genetic EMC algorithm and performs the mutation step.

Usage

```
mutation(chain, pop, pop.fit, force, data, fitness, t , impute = impute,
         b = NULL, a = 1, rec.mdl, burnin = TRUE)
```

Arguments

`chain`	integer specifying which chain (or model) of the population to perform the mutation on.
`pop`	matrix specifying the current status of each chain of the population.
`pop.fit`	vector of fitness values for each of the models specified in the current population.
`force`	character vector specify the variables to force in the models.
`data`	data frame of the same form in `Gene.EMC`
`fitness`	character string specifying the fitness function to use in the algorithm.
`t`	temperature vector specifying the temperature value for each chain in the population.
`impute`	number of imputed data sets.
`b`	If the fitness function is "AIC.BB", the user must specify the value for the beta hyper-parameter b.
`a`	If the fitness function is "AIC.BB", the user must specify the value for the beta hyper-parameter a.
`rec.mdl`	indicator vector that indicates which SNPs should not have a recessive parameter since the power is too weak.
`burnin`	integer indicating the length of the burnin.

Details

In the mutation step we are performing a Metropolis update on the population by choosing a model, or current value of one of the chains and taking one of the SNP indicators and mutating its state in the chosen model.

Value

This function outputs a list of the following values:

`samp`	current status of the chain of the current population.
`samp.fit`	fitness value for the model specified in the chain of the current population
`mut.a`	indicator if the mutation proposal was accpeted.

Author(s)

Melanie Wilson <maw27@stat.duke.edu>

References

Geyer C (1991). *Markov chain Monte Carlo maximum likelihood.* Computing Science and Statistics:156.

Holland J (1975). *Adaptation in Natural and Artificial Systems.* University of Michigan Press.

Liang F, Wong W (2000). *Evolutionary Monte Carol:Applications to Cp Model Sampling and Change Point Problem.* Statistica Sinica 10:317.

`post.prob` *Calculates Global and Marginal Summaries*

Description

This function calculates the global and marginal posterior probabilities and Bayes Factors that give the evidence of there being an association in the overall set of SNPs of interest, the individual genes of interest and the individual SNPs of interest.

Usage

```
post.prob(emc.out, sim.info, b = NULL, a = 1)
```

Arguments

`emc.out`	Output from `Gene.EMC`
`sim.info`	Vector of character strings giving the names of the genes for each of the SNPs in the model, or a data frame that includes a column labeled "SNP" and a column labeled "Gene". This data frame may contain all SNPs in the data, not just the ones in the model. SNPs must appear in the same order that they appear in the input data frame for Gene.EMC.
`b`	If the fitness function used in the Genetic EMC algorithm is "AIC.BB" the user must specify the value of b for the beta hyper-parameter.
`a`	If the fitness function used in the Genetic EMC algorithm is "AIC.BB" the user must specify the value of a for the beta hyper-parameter.

Details

Global and marginal summaries are computed based on calculating the posterior probabilities of each of the unique models that were visited in the Genetic EMC algorithm. The global summaries included a posterior probability of association in the overall set of SNPs and Bayes Factor for the hypothesis that there is an association in the overall set. The marginal summaries are calculated at the gene and the SNP level. At the gene level, posterior probabilities and Bayes Factors are computed for the overall evidence of at least one of the SNPs within the gene of interest being associated. At the SNP level, posterior probabilities and Bayes Factors are computed for the evidence of an association within the given SNP and posterior probabilities of the most likely genetic mode of inheritance of the SNP given that it is associated is computed (for the log-additive, dominant, and recessive models).

Value

The output of the function is a list of the following values:

`Post.Model`	the posterior probability of each of the unique models visited by the Genetic EMC algorithm
`BF.Assoc`	matrix with the global posterior probability and Bayes Factor of an overall association and prior odds of H0:Ha
`Post.SNP`	matrix where each row gives the gene name, inclusion probability, probability of log-additive, dominate, and recessive genetic mode of inheritance respectively, and the Bayes Factor for association for a given SNP.
`Post.Gene`	matrix where each row gives the inclusion probability and Bayes Factor for the evidence of an association for a given gene

Author(s)

Melanie Wilson <maw27@stat.duke.edu>

Examples

```
## Combine emc.out.1 and emc.out.2 to get results from both runs.
data(emc.out.1, emc.out.2)
emc.out <- combine.EMC(list(emc.out.1, emc.out.2))

data(sim.info)
p <- dim(emc.out.1$which)[2] - 1

##Calculate Posterior Quantities
post.prob.out <- post.prob(emc.out, sim.info, b=p)
```

`sim.info` *Associated Tag SNPs for Simulation*

Description

Matrix giving the information on the 60 SNPs in the `dna.snp.full` data set, giving the SNP rs number, the gene of the SNP, the assumed odds ratio of the SNP, the minor allele frequency of the SNP and the assumed genetic model of the SNP.

Usage

```
data(sim.info)
```

Examples

```
  data(sim.info)
```

Index