Package 'BEDASSLE'

February 19, 2015

Type Package

Title Quantifies effects of geo/eco distance on genetic differentiation

Version 1.5

Date 2014-12-03

Author Gideon Bradburd

Maintainer Gideon Bradburd <gbradburd@ucdavis.edu>

Description provides functions that allow users to quantify the relative contributions of geographic and ecological distances to empirical patterns of genetic differentiation on a landscape. Specifically, we use a custom Markov chain Monte Carlo (MCMC) algorithm, which is used to estimate the parameters of the inference model, as well as functions for performing MCMC diagnosis and assessing model adequacy.

License GPL (>= 2)

Imports MASS, matrixcalc, emdbook

NeedsCompilation no

Repository CRAN

Date/Publication 2014-12-04 00:30:44

R topics documented:

`BEDASSLE-package` *Disentangling the contributions of geographic and ecological isolation to genetic differentiation*

Description

This method models the covariance in allele frequencies between populations on a landscape as a decreasing function of their pairwise geographic and ecological distance. Allele frequencies are modeled as a spatial Gaussian process with a parametric covariance function. The parameters of this covariance function, as well as the spatially smoothed allele frequencies, are estimated in a custom Markov chain Monte Carlo.

Details

Package:	BEDASSLE
Type:	Package
Version:	1.5
Date:	2013-09-12
License:	GPL (>=2)

The two inference functions are `MCMC` and `MCMC_BB`, which call the Markov chain Monte Carlo algorithms on the standard and overdispersion (Beta-Binomial) models, respectively. To evaluate MCMC performance, there are a number of MCMC diagnosis and visualization functions, which variously show the trace, plots, marginal and joint marginal densities, and parameter acceptance rates. To evaluate model adequacy, there is a posterior predictive sample function (`posterior.predictive.sample`), and an accompanying function to plot its output and visually assess the model's ability to describe the user's data.

Author(s)

Gideon Bradburd

Maintainer: Gideon Bradburd <gbradburd@ucdavis.edu>

References

Bradburd, G.S., Ralph, P.L., and Coop, G.M. Disentangling the effects of geographic and ecological isolation on genetic differentiation. *Evolution* 2013.

`BEDASSLE-internal` *Internal BEDASSLE Functions*

Description

Internal BEDASSLE functions

Details

These functions are called by other functions (mostly `MCMC` and `MCMC_BB`), and will not be called directly by the user.

`calculate.all.pairwise.Fst` *Calculates unbiased pairwise Fst between all sampled populations*

Description

This function calculates unbiased F_{ST} (based on Weir and Hill's θ, 2002), between all populations/individuals included in the `counts` matrix, and returns the results in a `k` by `k` matrix, where `k = nrow(counts)`. Loci for which either of the populations/individuals has missing data (i.e. - the sample size is zero) are excluded.

Usage

```
calculate.all.pairwise.Fst(allele.counts, sample.sizes)
```

Arguments

`allele.counts`	A matrix of allelic count data, for which `nrow =` the number of populations and `ncol =` the number of bi-allelic loci sampled. Each cell gives the number of times allele '1' is observed in each population. The choice of which allele is allele '1' is arbitrary, but must be consistent across all populations at a locus.
`sample.sizes`	A matrix of sample sizes, for which `nrow =` the number of populations and `ncol =` the number of bi-allelic loci sampled (i.e. - the dimensions of `sample.sizes` must match those of `counts`). Each cell gives the number of chromosomes successfully genotyped at each locus in each population.

Value

A matrix of pairwise, unbiased F_{ST}.

Author(s)

Gideon Bradburd

References

Weir,B.S. and W.G. Hill. 2002. Estimating F-statistics. Ann.Rev.Gen. 36:949-952.

Examples

```
#With the HGDP dataset
data(HGDP.bedassle.data)

#Calculate pairwise Fst between all population pairs
hgdp.pairwise.Fst <- calculate.all.pairwise.Fst(
HGDP.bedassle.data$allele.counts,
HGDP.bedassle.data$sample.sizes
)

#Plot pairwise Fst against geographic distance
plot(HGDP.bedassle.data$GeoDistance,
hgdp.pairwise.Fst,
pch=19,
col=HGDP.bedassle.data$EcoDistance+1,
ylab="pairwise Fst",
xlab="geographic distance",
main="isolation by distance")
legend(x="bottomright",pch=19,col=c(1,2),
legend=c("same side of Himalayas",
"opposite sides of Himalayas"))
```

calculate.pairwise.Fst

Calculates unbiased pairwise Fst between a pair of populations

Description

This function calculates unbiased F_{ST} (based on Weir and Hill's θ, 2002), between a pair of populations/individuals. Loci for which either of the populations/individuals has missing data (i.e. - the sample size is zero) are excluded.

Usage

```
calculate.pairwise.Fst(allele.counts, sample.sizes)
```

Arguments

`allele.counts` A matrix of allele counts of dimensions `nrow = 2`, one for each of the two populations/individuals between which pairwise F_{ST} is being calculated, and `ncol =` the number of loci genotyped.

`sample.sizes` A matrix of sample sizes of dimensions `nrow = 2`, one for each of the two populations/individuals between which pairwise F_{ST} is being calculated, and `ncol =` the number of loci genotyped (i.e. - the dimensions of `sample.sizes` must match those of `counts`). Each cell gives the number of chromosomes successfully genotyped at each locus in each population.

Value

Pairwise unbiased F_{ST} between a pair of populations/individuals

Author(s)

Gideon Bradburd

References

Weir,B.S. and W.G. Hill. 2002. Estimating F-statistics. Ann.Rev.Gen. 36:949-952.

Examples

```
#With the HGDP dataset
data(HGDP.bedassle.data)

#Draw 2 populations at random from the Eurasian HGDP dataset
pop1 <- sample(nrow(HGDP.bedassle.data$allele.counts),1)
pop2 <- sample(nrow(HGDP.bedassle.data$allele.counts),1)

#Calculate unbiased Fst between them
pairwise.Fst <- calculate.pairwise.Fst(
HGDP.bedassle.data$allele.counts[c(pop1,pop2),],
HGDP.bedassle.data$sample.sizes[c(pop1,pop2),]
)

#Print that Fst to the console
print(sprintf("Fst between the %s population and the %s population is %s",
HGDP.bedassle.data$hgdp.metadata[pop1,1],
HGDP.bedassle.data$hgdp.metadata[pop2,1],
round(pairwise.Fst,3))
 )
```

Covariance *The parametric covariance matrix*

Description

This function parameterizes the decay in covariance of transformed allele frequencies between sampled populations/individuals over their pairwise geographic and ecological distance.

Usage

```
Covariance(a0, aD, aE, a2, GeoDist, EcoDist, delta)
```

Arguments

a0	This parameter controls the variance when pairwise distance is zero. It is the variance of the population-specific transformed allelic deviate (theta) when pairwise distances are zero (i.e. when $D_{i,j} + E_{i,j} = 0$).
aD	This parameter gives the effect size of geographic distance ($D_{i,j}$).
aE	This parameter gives the effect size(s) of ecological distance(s) ($E_{i,j}$).
a2	This parameter controls the shape of the decay in covariance with distance.
GeoDist	Pairwise geographic distance ($D_{i,j}$). This may be Euclidean, or, if the geographic scale of sampling merits it, great-circle distance.
EcoDist	Pairwise ecological distance(s) ($E_{i,j}$), which may be continuous (e.g. - difference in elevation) or binary (same or opposite side of some hypothesized barrier to gene flow).
delta	This gives the size of the "delta shift" on the off-diagonal elements of the parametric covariance matrix, used to ensure its positive-definiteness (even, for example, when there are separate populations sampled at the same geographic/ecological coordinates). This value must be large enough that the covariance matrix is positive-definite, but, if possible, should be smaller than the smallest off-diagonal distance elements, lest it have an undue impact on inference. If the user is concerned that the delta shift is too large relative to the pairwise distance elements in D and E, she should run subsequent analyses, varying the size of delta, to see if it has an impact on model inference.

Author(s)

Gideon Bradburd

Examples

```
#With the HGDP dataset
data(HGDP.bedassle.data)

#Draw random values of the {alpha} parameters from their priors
alpha0 <- rgamma(1,shape=1,rate=1)
```

```
alphaD <- rexp(1,rate=1)
alphaE <- matrix(rexp(1,rate=1),nrow=1,ncol=1)
alpha2 <- runif(1,0.1,2)

#Parameterize the covariance function using the HGDP dataset distances (Geo and Eco)
example.covariance <- Covariance(a0 = alpha0,aD = alphaD,aE = alphaE,a2 = alpha2,
GeoDist = HGDP.bedassle.data$GeoDistance,
EcoDist = list(HGDP.bedassle.data$EcoDistance),
delta = 0.001)

#Plot the example covariance against geographic distance
plot(HGDP.bedassle.data$GeoDistance,
example.covariance,
pch=19,col=HGDP.bedassle.data$EcoDistance+1,
main="Covariance in allele frequencies across the Himalayas")
legend(x="topright",pch=19,col=c(1,2),
legend=c("same side of Himalayas",
"opposite sides of Himalayas"))
```

HGDP.bedassle.data *The Eurasian subset of the HGDP dataset used in example BEDASSLE analyses*

Description

The allelic counts, sample sizes, geographic distances, ecological distances, and population metadata from the 38 human populations used in example BEDASSLE analyses, subsetted from the Human Genome Diversity Panel (HGDP) dataset.

Usage

```
data(HGDP.bedassle.data)
```

Format

The format is: List of 7

$ allele.counts : int [1:38, 1:1000] 12 16 5 17 4 14 20 5 34 ...

- ..- attr(*, "dimnames")=List of 2
- .. ..$: chr [1:38] "Adygei" "Basque" "Italian" "French" ...
- .. ..$: chr [1:1000] "rs13287637" "rs17792496" "rs1968588" ...

$ sample.sizes : int [1:38, 1:1000] 34 48 24 56 30 50 56 ...

- ..- attr(*, "dimnames")=List of 2
- .. ..$: chr [1:38] "Adygei" "Basque" "Italian" "French" ...
- .. ..$: chr [1:1000] "rs13287637" "rs17792496" "rs1968588" ...

$ GeoDistance : num [1:38, 1:38] 0 1.187 0.867 1.101 1.247 ...

$ EcoDistance : num [1:38, 1:38] 0 0 0 0 0 0 0 0 0 0 ...

- ..- attr(*, "dimnames")=List of 2
- .. ..$: chr [1:38] "1" "2" "3" "4" ...
- .. ..$: chr [1:38] "1" "2" "3" "4" ...

$ number.of.populations: int 38

$ number.of.loci : int 1000

$ hgdp.metadata : 'data.frame': 38 obs. of 3 variables:

..$ Population: chr [1:38] "Adygei" "Basque" "Italian" ...

..$ Latitude : chr [1:38] "44" "43" "46" "46" ...

..$ Longitude : chr [1:38] "39" "0" "10" "2" ...

Details

allele.counts A matrix of allelic count data, for which `nrow =` the number of populations and `ncol =` the number of bi-allelic loci sampled. Each cell gives the number of times allele '1' is observed in each population. The choice of which allele is allele '1' is arbitrary, but must be consistent across all populations at a locus.

sample.sizes A matrix of sample sizes, for which `nrow =` the number of populations and `ncol =` the number of bi-allelic loci sampled (i.e. - the dimensions of `sample.sizes` must match those of `counts`). Each cell gives the number of chromosomes successfully genotyped at each locus in each population.

Geo.Distance Pairwise geographic distance ($D_{i,j}$). This may be Euclidean, or, if the geographic scale of sampling merits it, great-circle distance. In the case of this dataset, it is great-circle distance.

Eco.Distance Pairwise ecological distance(s) ($E_{i,j}$), which may be continuous (e.g. - difference in elevation) or binary (same or opposite side of some hypothesized barrier to gene flow). In this case, the ecological distance is binary, representing whether a pair of populations occurs on the same side, or on opposite sides, of the Himalayas.

number.of.populations The number of populations in the analysis. This should be equal to `nrow(counts)`. In this dataset, there are 38 populations sampled.

number.of.loci The number of loci in the analysis. This should be equal to `ncol(counts)`. In this dataset, there are 1000 loci sampled.

hgdp.metadata This data frame contains the metadata on the populations included in the analysis, including:

- Population name
- Latitude
- Longitude

Source

- Conrad et al. A worldwide survey of haplotype variation and linkage disequilibrium in the human genome. *Nature Genetics* 2008.
- Li et al. Worldwide human relationships inferred from genome-wide patterns of variation. *Science* 2008.

References

Bradburd, G.S., Ralph, P.L., and Coop, G.M. Disentangling the effects of geographic and ecological isolation on genetic differentiation. *Evolution* 2013.

Examples

```
## see \command{MCMC}, \command{MCMC_BB}, \command{calculate.pariwise.Fst},
## \command{calculate.all.pairwise.Fst}, and \command{Covariance} for usage.
```

`link.up.posteriors` *Links up multiple MCMC output objects*

Description

Creates a single MCMC output object that links together the output from 2 different runs. To be used when analyses are run serially on a single dataset, with subsequent runs initiated at the parameter values estimated in the last generation of the previous MCMC run.

Usage

```
link.up.posteriors(MCMC.output1, MCMC.output2, linked.up.output.file.name)
```

Arguments

`MCMC.output1` The first (chronologically) MCMC output to be incorporated into the linked-up output object.

`MCMC.output2` The second (chronologically) MCMC output to be incorporated into the linked-up output object.

`linked.up.output.file.name`
The linked-up MCMC object file name. The suffix ".Robj" will be appended to the user-specified name.

Details

Acceptance rates are re-calculated to be consistent across the new, larger MCMC object. The function is also flexible with respect to the model parameterization (e.g. - it will recognize, for example, whether users have specified the standard or beta-binomial models, or whether users have specified one, or more than one, alphaE parameters).

Author(s)

Gideon Bradburd

make.continuing.params *Generates an R object containing the last parameter values of an MCMC run (to be used for a subsequent run)*

Description

This function creates an R object that contains the parameter values read from the last generation of a previous MCMC run. This R object can then be used to initiate a subsequent analysis, effectively creating a single long chain. [A single MCMC object from both runs can be created using the function `link.up.posteriors`].

Usage

```
make.continuing.params(MCMC.output, file.name)
```

Arguments

`MCMC.output`	The standard MCMC output file generated from a BEDASSLE run.
`file.name`	The user-defined name assigned to the R object of parameters to be used in continuing analysis.

Author(s)

Gideon Bradburd

MCMC *Runs the Markov chain Monte Carlo with the standard (Binomial) model*

Description

This function initiates the Markov chain Monte Carlo (MCMC) for the binomial BEDASSLE model.

Usage

```
MCMC(counts, sample_sizes, D, E, k, loci, delta, aD_stp, aE_stp, a2_stp, thetas_stp,
mu_stp, ngen, printfreq, savefreq, samplefreq, directory = NULL, prefix = "",
continue = FALSE, continuing.params = NULL)
```

Arguments

`counts` A matrix of allelic count data, for which `nrow =` the number of populations and `ncol =` the number of bi-allelic loci sampled. Each cell gives the number of times allele ‘1’ is observed in each population. The choice of which allele is allele ‘1’ is arbitrary, but must be consistent across all populations at a locus.

`sample_sizes` A matrix of sample sizes, for which `nrow =` the number of populations and `ncol =` the number of bi-allelic loci sampled (i.e. - the dimensions of `sample.sizes` must match those of `counts`). Each cell gives the number of chromosomes successfully genotyped at each locus in each population.

`D` Pairwise geographic distance ($D_{i,j}$). This can be two-dimensional Euclidean distance, or great-circle distance, or, in fact, any positive definite matrix (deriving, for instance, from a resistance distance). However, note that the algorithm silently restricts the prior on the alpha parameters, and specifically the alpha_2 parameter, to the part of parameter space that results in valid covariance matrices; in the case of two-dimensional Euclidean distances, this will not happen, since any value of alpha_2 between 0 and 2 is valid (see Guillot et al.’s "Valid covariance models for the analysis of geographical genetic variation" for more detail on this).

`E` Pairwise ecological distance(s) ($E_{i,j}$), which may be continuous (e.g. - difference in elevation) or binary (same or opposite side of some hypothesized barrier to gene flow). Users may specify one or more ecological distance matrices. If more than one is specified, they should be formatted as a `list`.

`k` The number of populations in the analysis. This should be equal to `nrow(counts)`.

`loci` The number of loci in the analysis. This should be equal to `ncol(counts)`

`delta` The size of the "delta shift" on the off-diagonal elements of the parametric covariance matrix, used to ensure its positive-definiteness (even, for example, when there are separate populations sampled at the same geographic/ecological coordinates). This value must be large enough that the covariance matrix is positive-definite, but, if possible, should be smaller than the smallest off- diagonal distance elements, lest it have an undue impact on inference. If the user is concerned that the delta shift is too large relative to the pairwise distance elements in `D` and `E`, she should run subsequent analyses, varying the size of delta, to see if it has an impact on model inference.

`aD_stp` The scale of the tuning parameter on aD (alphaD). The scale of the tuning parameter is the standard deviation of the normal distribution from which small perturbations are made to those parameters updated via a random-walk sampler. A larger value of the scale of the tuning parameter will lead to, on average, larger proposed moves and lower acceptance rates (for more on acceptance rates, see `plot_acceptance_rate`).

`aE_stp` The scale of the tuning parameter on aE (alphaE). If there are multiple ecological distances included in the analysis, there will be multiple alphaE parameters (one for each matrix in the list of E). These may be updated all with the same scale of a tuning parameter, or they can each get their own, in which case aE_stp should be a vector of length equal to the number of ecological distance variables.

`a2_stp` The scale of the tuning parameter on a2 (alpha_2).

`thetas_stp` The scale of the tuning parameter on the theta parameters.

`mu_stp` The scale of the tuning parameter on mu.

`ngen` The number of generations over which to run the MCMC (one parameter is updated at random per generation, with mu, theta, and phi all counting, for the purposes of updates, as one parameter).

`printfreq` The frequency with which MCMC progress is printed to the screen. If `printfreq =1000`, an update with the MCMC generation number and the posterior probability at that generation will print to the screen every 1000 generations.

`savefreq` The frequency with which the MCMC saves its output as an R object (`savefreq = 50,000` means that MCMC output is saved every 50,000 generations). If `ngen` is large, this saving process may be computationally expensive, and so should not be performed too frequently. However, users may wish to evalute MCMC performance while the chain is still running, or may be forced to truncate runs early, and should therefore specify a `savefreq` that is less than `ngen`. We recommend a `savefreq` of between 1/10th and 1/20th of `ngen`.

`samplefreq` The thinning of the MCMC chain (`samplefreq = 1000` means that the parameter values saved in the MCMC output are sampled once every 1000 generations). A higher `samplefreq` will decrease parameter autocorrelation time. However, there is still information in autocorrelated draws from the joint posterior, so the `samplefreq` should be viewed merely as a computational convenience, to decrease the size of the MCMC output objects.

`directory` If specified, this points to a directory into which output will be saved.

`prefix` If specified, this prefix will be added to all output file names.

`continue` If `TRUE`, this will initiate the MCMC chain from the last parameter values of a previous analysis. This option can be used to effectively increase the `ngen` of an initial run. If `FALSE`, the MCMC will be initiated from random parameter values.

`continuing.params`
The list of parameter values used to initiate the MCMC if `continue = TRUE`. If the user wants to continue an analysis on a dataset, these should be the parameter values from the last generation of the previous analysis. This list may be generated using the function `make.continuing.params`.

Details

This function saves an MCMC output object at intervals specified by `savefreq`. This object may be ported into R working memory using the *base* function `load`.

As with any MCMC method, it is very important here to perform MCMC diagnosis and evaluate chain mixing. I have provided a number of MCMC diagnosis graphing functions for user convenience in visually assessing MCMC output. These include `plot_all_trace`;`plot_all_marginals`; `plot_all_joint_marginals`; and `plot_all_acceptance_rates`. To evaluate model adequacy, users should use `posterior.predictive.sample` and `plot_posterior_predictive_sample`. These MCMC diagnosis/model adequacy functions all call the standard MCMC output R object that the BEDASSLE MCMC generates as their principal argument.

If users wish to start another MCMC run from where the current run left off, they should use `make.continuing.params`, and initiate the new run with option `continue = TRUE` and the `continuing.params` list from the previous run specified.

Author(s)

Gideon Bradburd

References

Bradburd, G.S., Ralph, P.L., and Coop, G.M. Disentangling the effects of geographic and ecological isolation on genetic differentiation. *Evolution* 2013.

Examples

```
#With the HGDP dataset and mcmc operators
data(HGDP.bedassle.data)
data(mcmc.operators)

#The value of delta may set off warnings,
#so temporarily disable warnings.
op <- options("warn")
options(warn = -1)

#Call the Markov chain Monte Carlo for the standard model
MCMC(
counts = HGDP.bedassle.data$allele.counts,
sample_sizes = HGDP.bedassle.data$sample.sizes,
D = HGDP.bedassle.data$GeoDistance,
E = HGDP.bedassle.data$EcoDistance,
k = HGDP.bedassle.data$number.of.populations,
loci = HGDP.bedassle.data$number.of.loci,
delta = mcmc.operators$delta,
aD_stp = mcmc.operators$aD_stp,
aE_stp = mcmc.operators$aE_stp,
a2_stp = mcmc.operators$a2_stp,
thetas_stp = mcmc.operators$thetas_stp,
mu_stp = mcmc.operators$mu_stp,
ngen = mcmc.operators$ngen,
printfreq = mcmc.operators$printfreq,
savefreq = mcmc.operators$savefreq,
samplefreq = mcmc.operators$samplefreq,
directory = NULL,
prefix = mcmc.operators$prefix,
continue = FALSE,
continuing.params = NULL
)

#Re-enable warnings
options(op)
```

`mcmc.operators` *Operator parameters that control the operation of the MCMC*

Description

These parameters, which are passed to the `MCMC` and `MCMC_BB` functions, control the operation of the MCMC. They specify the number of generations over which the MCMC runs; the scales of the tuning parameters (stp) for all parameters updated via random-walk samplers; the save, print, and sample frequency of the chain, and the output file names.

Usage

```
data(mcmc.operators)
```

Format

The format is: List of 12

$ delta : num 0.001

$ aD_stp : num 0.0018

$ aE_stp : num 0.04

$ a2_stp : num 0.0035

$ phi_stp : num 30

$ thetas_stp: num 0.07

$ mu_stp : num 0.17

$ ngen : num 100

$ printfreq : num 2

$ savefreq : num 100

$ samplefreq: num 5

$ prefix : chr "example_"

Details

delta The size of the "delta shift" on the off-diagonal elements of the parametric covariance matrix, used to ensure its positive-definiteness (even, for example, when there are separate populations sampled at the same geographic/ecological coordinates). This value must be large enough that the covariance matrix is positive-definite, but, if possible, should be smaller than the smallest off-diagonal distance elements, lest it have an undue impact on inference. If the user is concerned that the delta shift is too large relative to the pairwise distance elements in `D` and `E`, she should run subsequent analyses, varying the size of delta, to see if it has an impact on model inference.

aD_stp The scale of the tuning parameter on aD (alphaD). The scale of the tuning parameter is the standard deviation of the normal distribution from which small perturbations are made to those parameters updated via a random-walk sampler. A larger value of the scale of the tuning parameter will lead to, on average, larger proposed moves and lower acceptance rates (for more on acceptance rates, see `plot_acceptance_rate`).

aE_stp The scale of the tuning parameter on aE (alphaE). If there are multiple ecological distances included in the analysis, there will be multiple alphaE parameters (one for each matrix in the list of E). These may be updated all with the same scale of a tuning parameter, or they can each get their own, in which case aE_stp should be a vector of length equal to the number of ecological distance variables.

a2_stp The scale of the tuning parameter on a2 (alpha_2).

phi_stp The scale of the tuning parameter on the phi parameters.

thetas_stp The scale of the tuning parameter on the theta parameters.

mu_stp The scale of the tuning parameter on mu.

ngen The number of generations over which to run the MCMC (one parameter is updated at random per generation, with mu, theta, and phi all counting, for the purposes of updates, as one parameter).

printfreq The frequency with which MCMC progress is printed to the screen. If `printfreq =1000`, an update with the MCMC generation number and the posterior probability at that generation will print to the screen every 1000 generations.

savefreq The frequency with which the MCMC saves its output as an R object (`savefreq =50,000` means that MCMC output is saved every 50,000 generations). If `ngen` is large, this saving process may be computationally expensive, and so should not be performed too frequently. However, users may wish to evalute MCMC performance while the chain is still running, or may be forced to truncate runs early, and should therefore specify a `savefreq` that is less than `ngen`. We recommend a `savefreq` of between 1/10th and 1/20th of `ngen`.

samplefreq The thinning of the MCMC chain (`samplefreq = 1000` means that the parameter values saved in the MCMC output are sampled once every 1000 generations). A higher `samplefreq` will decrease parameter autocorrelation time. However, there is still information in autocorrelated draws from the joint posterior, so the `samplefreq` should be viewed merely as a computational convenience, to decrease the size of the MCMC output objects.

prefix If specified, this prefix will be added to all output file names.

Examples

```
## see \command{MCMC} and \command{MCMC_BB} for example usage.
```

`MCMC_BB` *Runs the Markov chain Monte Carlo with the overdispersion (Beta-Binomial) model*

Description

This function initiates the Markov chain Monte Carlo (MCMC) for the beta-binomial BEDASSLE model. The beta-binomial model allows populations to diverge from the model's expectations based on their location and their neighbors.

Usage

```
MCMC_BB(counts, sample_sizes, D, E, k, loci, delta, aD_stp, aE_stp, a2_stp, phi_stp,
thetas_stp, mu_stp, ngen, printfreq, savefreq, samplefreq, directory = NULL, prefix = "",
continue = FALSE, continuing.params = NULL)
```

Arguments

`counts` A matrix of allelic count data, for which `nrow =` the number of populations and `ncol =` the number of bi-allelic loci sampled. Each cell gives the number of times allele '1' is observed in each population. The choice of which allele is allele '1' is arbitrary, but must be consistent across all populations at a locus.

`sample_sizes` A matrix of sample sizes, for which `nrow =` the number of populations and `ncol =` the number of bi-allelic loci sampled (i.e. - the dimensions of `sample.sizes` must match those of `counts`). Each cell gives the number of chromosomes successfully genotyped at each locus in each population.

`D` Pairwise geographic distance ($D_{i,j}$). This can be two-dimensional Euclidean distance, or great-circle distance, or, in fact, any positive definite matrix (deriving, for instance, from a resistance distance). However, note that the algorithm silently restricts the prior on the alpha parameters, and specifically the alpha_2 parameter, to the part of parameter space that results in valid covariance matrices; in the case of two-dimensional Euclidean distances, this will not happen, since any value of alpha_2 between 0 and 2 is valid (see Guillot et al.'s "Valid covariance models for the analysis of geographical genetic variation" for more detail on this).

`E` Pairwise ecological distance(s) ($E_{i,j}$), which may be continuous (e.g. - difference in elevation) or binary (same or opposite side of some hypothesized barrier to gene flow). Users may specify one or more ecological distance matrices. If more than one is specified, they should be formatted as a `list`.

`k` The number of populations in the analysis. This should be equal to `nrow(counts)`.

`loci` The number of loci in the analysis. This should be equal to `ncol(counts)`

`delta` The size of the "delta shift" on the off-diagonal elements of the parametric covariance matrix, used to ensure its positive-definiteness (even, for example, when there are separate populations sampled at the same geographic/ecological coordinates). This value must be large enough that the covariance matrix is positive-definite, but, if possible, should be smaller than the smallest off- diagonal distance elements, lest it have an undue impact on inference. If the user is concerned that the delta shift is too large relative to the pairwise distance elements in `D` and `E`, she should run subsequent analyses, varying the size of delta, to see if it has an impact on model inference.

`aD_stp` The scale of the tuning parameter on aD (alphaD). The scale of the tuning parameter is the standard deviation of the normal distribution from which small perturbations are made to those parameters updated via a random-walk sampler. A larger value of the scale of the tuning parameter will lead to, on average, larger proposed moves and lower acceptance rates (for more on acceptance rates, see `plot_acceptance_rate`).

`aE_stp` The scale of the tuning parameter on aE (alphaE). If there are multiple ecological distances included in the analysis, there will be multiple alphaE parameters (one for each matrix in the list of E). These may be updated all with the same scale of a tuning parameter, or they can each get their own, in which case aE_stp should be a vector of length equal to the number of ecological distance variables.

`a2_stp` The scale of the tuning parameter on a2 (alpha_2).

`phi_stp` The scale of the tuning parameter on the phi parameters.

`thetas_stp` The scale of the tuning parameter on the theta parameters.

`mu_stp` The scale of the tuning parameter on mu.

`ngen` The number of generations over which to run the MCMC (one parameter is updated at random per generation, with mu, theta, and phi all counting, for the purposes of updates, as one parameter).

`printfreq` The frequency with which MCMC progress is printed to the screen. If `printfreq =1000`, an update with the MCMC generation number and the posterior probability at that generation will print to the screen every 1000 generations.

`savefreq` The frequency with which the MCMC saves its output as an R object (`savefreq = 50,000` means that MCMC output is saved every 50,000 generations). If `ngen` is large, this saving process may be computationally expensive, and so should not be performed too frequently. However, users may wish to evalute MCMC performance while the chain is still running, or may be forced to truncate runs early, and should therefore specify a `savefreq` that is less than `ngen`. We recommend a `savefreq` of between 1/10th and 1/20th of `ngen`.

`samplefreq` The thinning of the MCMC chain (`samplefreq = 1000` means that the parameter values saved in the MCMC output are sampled once every 1000 generations). A higher `samplefreq` will decrease parameter autocorrelation time. However, there is still information in autocorrelated draws from the joint posterior, so the `samplefreq` should be viewed merely as a computational convenience, to decrease the size of the MCMC output objects.

`directory` If specified, this points to a directory into which output will be saved.

`prefix` If specified, this prefix will be added to all output file names.

`continue` If `TRUE`, this will initiate the MCMC chain from the last parameter values of a previous analysis. This option can be used to effectively increase the `ngen` of an initial run. If `FALSE`, the MCMC will be initiated from random parameter values.

`continuing.params`
The list of parameter values used to initiate the MCMC if `continue = TRUE`. If the user wants to continue an analysis on a dataset, these should be the parameter values from the last generation of the previous analysis. This list may be generated using the function `make.continuing.params`.

Details

This function saves an MCMC output object at intervals specified by `savefreq`. This object may be ported into R working memory using the *base* function `load`.

As with any MCMC method, it is very important here to perform MCMC diagnosis and evaluate chain mixing. I have provided a number of MCMC diagnosis graphing functions for user convenience in visually assessing MCMC output. These include plot_all_trace;plot_all_marginals; plot_all_joint_marginals; and plot_all_acceptance_rates. To evaluate model adequacy, users should use posterior.predictive.sample and plot_posterior_predictive_sample. These MCMC diagnosis/model adequacy functions all call the standard MCMC output R object that the BEDASSLE MCMC generates as their principal argument.

If users wish to start another MCMC run from where the current run left off, they should use make.continuing.params, and initiate the new run with option continue = TRUE and the continuing.params list from the previous run specified.

Author(s)

Gideon Bradburd

References

Bradburd, G.S., Ralph, P.L., and Coop, G.M. Disentangling the effects of geographic and ecological isolation on genetic differentiation. *Evolution* 2013.

Examples

```
#With the HGDP dataset and mcmc operators
data(HGDP.bedassle.data)
data(mcmc.operators)

#The beta-binomial likelihood function may generate "NaNs produced" warnings,
#so temporarily disable warnings.
op <- options("warn")
options(warn = -1)

#Call the Markov chain Monte Carlo for the overdispersion model
MCMC_BB(
counts = HGDP.bedassle.data$allele.counts,
sample_sizes = HGDP.bedassle.data$sample.sizes,
D = HGDP.bedassle.data$GeoDistance,
E = HGDP.bedassle.data$EcoDistance,
k = HGDP.bedassle.data$number.of.populations,
loci = HGDP.bedassle.data$number.of.loci,
delta = mcmc.operators$delta,
aD_stp = mcmc.operators$aD_stp,
aE_stp = mcmc.operators$aE_stp,
a2_stp = mcmc.operators$a2_stp,
phi_stp = mcmc.operators$phi_stp,
thetas_stp = mcmc.operators$thetas_stp,
mu_stp = mcmc.operators$mu_stp,
ngen = mcmc.operators$ngen,
printfreq = mcmc.operators$printfreq,
savefreq = mcmc.operators$savefreq,
samplefreq = mcmc.operators$samplefreq,
directory = NULL,
prefix = mcmc.operators$prefix,
continue = FALSE,
continuing.params = NULL
)

#Re-enable warnings
options(op)
```

plot_acceptance_rate *Plots the acceptance rate of a parameter across MCMC generations*

Description

Creates a plot showing the proportion of proposed moves to accepted moves over the duration of the MCMC analysis.

Usage

```
plot_acceptance_rate(accepted.moves, proposed.moves, param.name =
deparse(substitute(accepted.moves)))
```

Arguments

`accepted.moves`	A vector giving the number of accepted random-walk moves at each sampled MCMC generation.
`proposed.moves`	A vector giving the number of proposed random-walk moves at each sampled MCMC generation.
`param.name`	The name of the parameter for which the trace plot is being displayed.

Details

For optimal mixing, between ~20 samplers should be accepted. If the acceptance rates fall outside that range, this function will automatically highlight that parameter as a potential instance of poor mixing. If the acceptance rates are too low, then for subsequent analyses the user should *decrease* the scale of the tuning parameter (or "std," as in, e.g., "aD_std"), and if acceptance rates are too high, the user should *increase* the scale of the tuning parameter. The scale of the tuning parameter is the standard deviation of the normal distribution from which the small random variable is drawn and added to the current parameter value to propose a move. If the acceptance rate has not plateaued by the end of an analysis, it is an indication that the chain may still be "going somewhere" in parameter space, and subsequent analyses should be performed.

Author(s)

Gideon Bradburd

plot_all_acceptance_rates *Plots the acceptance rates of all parameters across MCMC generations*

Description

Creates a series of plots showing the proportion of proposed moves to accepted moves over the duration of the MCMC analysis for each parameter updated via a random-walk sampler.

Usage

```
plot_all_acceptance_rates(MCMC.output)
```

Arguments

MCMC.output	The standard MCMC output file generated from a BEDASSLE run.

Details

For optimal mixing, between ~20 samplers should be accepted. If the acceptance rates fall outside that range, this function will automatically highlight that parameter as a potential instance of poor mixing. If the acceptance rates are too low, then for subsequent analyses the user should *decrease* the scale of the tuning parameter (or "std," as in, e.g., aD_std), and if acceptance rates are too high, the user should *increase* the scale of the tuning parameter. The scale of the tuning parameter is the standard deviation of the normal distribution from which the small random variable is drawn and added to the current parameter value to propose a move. If the acceptance rate has not plateaued by the end of an analysis, it is an indication that the chain may still be "going somewhere" in parameter space, and subsequent analyses should be performed.

Author(s)

Gideon Bradburd

plot_all_joint_marginals *Plots the joint marginals for all parameter pairs*

Description

For each sampled MCMC generation, the values estimated for a pair of parameters are logged and plotted against one another. Points are color coded by when in the analysis they were sampled, so that users can visually assess mixing. A joint marginal plot is generated for all combinations of parameters, excluding the phi parameters estimated in the beta-binomial model.

Usage

```
plot_all_joint_marginals(MCMC.output, percent.burnin = 0, thinning = 1)
```

Arguments

`MCMC.output`	The standard MCMC output file generated from a BEDASSLE run.
`percent.burnin`	The percent of the sampled MCMC generations to be discarded as "burn-in." If the MCMC is run for 1,000,000 generations, and sampled every 1,000 generations, there will be 1,000 sampled generations. A `percent.burnin` of `20` will discard the first 200 sampled parameter values from that sample.
`thinning`	The multiple by which the sampled MCMC generations are thinned. A `thinning` of `5` will sample every 5th MCMC generation.

Details

Visualizations of the joint marginal distributions allow users to (1) assess how well the MCMC is mixing, and (2) potentially diagnose instances of non-identifiability in the model. Strong linear trends in the joint marginal, or visible "ridges" in the likelihood surface, may be indicative of parameter non-identifiability, in which multiple combinations of values of these two parameters provide equally reasonable fits to the data.

Author(s)

Gideon Bradburd

`plot_all_marginals` *Plots the marginal densities for all parameters*

Description

Plots the posterior marginal density of all parameters. Users may specify whether they want a histogram, a density, or both.

Usage

```
plot_all_marginals(MCMC.output, percent.burnin = 0, thinning = 1,
population.names = NULL)
```

Arguments

`MCMC.output`	The standard MCMC output file generated from a BEDASSLE run.
`percent.burnin`	The percent of the sampled MCMC generations to be discarded as "burn-in." If the MCMC is run for 1,000,000 generations, and sampled every 1,000 generations, there will be 1,000 sampled generations. A `percent.burnin` of `20` will discard the first 200 sampled parameter values from that sample.

`thinning` The multiple by which the sampled MCMC generations are thinned. A `thinning` of `5` will sample every 5th MCMC generation.

`population.names` A vector of length `k`, where `k` is the number of populations/individuals (i.e. `k = nrow(counts)`), giving the name or identifier of each population/individual included in the analysis. These will be used to title the `k` marginal plots of the phi parameters estimated for each population/individual in the beta-binomial model. If the binomial model is used, `population.names` will not be used by this function.

Details

The marginal plot is another basic visual tool for MCMC diagnosis. Users should look for marginal plots that are "smooth as eggs" (indicating that the chain has been run long enough) and unimodal (indicating a single peak in the likelihood surface).

Author(s)

Gideon Bradburd

`plot_all_phi_marginals` *Plot all the marginals for the phi parameters for all populations*

Description

Plots the posterior marginal densities of all phi parameters. Users may specify whether they want a histogram, a density, or both. For convenience, the F_k statistic is presented in place of the phi parameter, as this is the statistic users care about. F_k is defined as $\frac{1}{1+phi_k}$.

Usage

```
plot_all_phi_marginals(phi_mat, percent.burnin = 0, thinning = 1,
population.names = NULL, pop.index= NULL, histogram = TRUE, density = TRUE)
```

Arguments

`phi_mat` The `k` by `ngen` matrix of phi values estimated for all `k` populations/individuals included in the analysis in each of `ngen` MCMC generations.

`percent.burnin` The percent of the sampled MCMC generations to be discarded as "burn-in." If the MCMC is run for 1,000,000 generations, and sampled every 1,000 generations, there will be 1,000 sampled generations. A `percent.burnin` of `20` will discard the first 200 sampled parameter values from that sample.

`thinning` The multiple by which the sampled MCMC generations are thinned. A `thinning` of `5` will sample every 5th MCMC generation.

`population.names`
A vector of length `k`, where `k` is the number of populations/individuals (i.e. `k = nrow(counts)`), giving the name or identifier of each population/individual included in the analysis. These will be used to title the `k` trace plots of the phi parameters estimated for each population/individual in the beta- binomial model. If `population.names` is not provided (i.e. `population.names = NULL`), a population index number will be used to title the plot.

`pop.index` A population index number generated to title a marginal plot if no `population.names` is specified.

`histogram` A switch that controls whether or not the plot contains a histogram of the values estimated for the parameter over the course of the MCMC. Default is `TRUE`.

`density` A switch that controls whether or not the plot shows the density of the values estimated for the parameter over the course of the MCMC. Default is `TRUE`.

Details

The marginal plot is another basic visual tool for MCMC diagnosis. Users should look for marginal plots that are "smooth as eggs" (indicating that the chain has been run long enough) and unimodal (indicating a single peak in the likelihood surface).

Author(s)

Gideon Bradburd

`plot_all_phi_trace` *Plots all the trace plots for the phi parameters for all populations*

Description

Plots all trace plots for the phi parameters in all populations. For convenience, the F_k statistic is presented in place of the phi parameter, as this is the statistic users care about. F_k is defined as $\frac{1}{1+phi_k}$.

Usage

```
plot_all_phi_trace(phi_mat, percent.burnin = 0, thinning = 1, population.names = NULL)
```

Arguments

`phi_mat` The `k` by `ngen` matrix of phi values estimated for all `k` populations/individuals included in the analysis in each of `ngen` MCMC generations.

`percent.burnin` The percent of the sampled MCMC generations to be discarded as "burn-in." If the MCMC is run for 1,000,000 generations, and sampled every 1,000 generations, there will be 1,000 sampled generations. A `percent.burnin` of `20` will discard the first 200 sampled parameter values from that sample.

`thinning` The multiple by which the sampled MCMC generations are thinned. A `thinning` of `5` will sample every 5th MCMC generation.

`population.names`
: A vector of length `k`, where `k` is the number of populations/individuals (i.e. `k = nrow(counts)`), giving the name or identifier of each population/individual included in the analysis. These will be used to title the `k` trace plots of the phi parameters estimated for each population/individual in the beta-binomial model. If `population.names` is not provided (i.e. `population.names =NULL`), a population index number will be used to title the plot.

Details

A trace plot is a basic visual tool for assessing MCMC mixing. If the chain is mixing well, the trace plot will resemble a "fuzzy caterpillar." If the trace plot has not plateaued, it is an indication that the chain has not converged on the stationary posterior distribution, and must be run longer. If the trace plot of a parameter exhibits high autocorrelation, the user may wish to either increase or decrease the scale of the tuning parameter on that parameter, to decrease or increase acceptance rates, respectively. If the chain appears to be bouncing between areas of "fuzzy caterpillar-dom," it may be an indication of a multi-modal likelihood surface.

Author(s)

Gideon Bradburd

`plot_all_trace` *Plots all the trace plots for all parameters*

Description

This function plots the parameter value estimated in each sampled generation of the MCMC against the index of that sampled generation for each parameter in the model.

Usage

```
plot_all_trace(MCMC.output, percent.burnin = 0, thinning = 1, population.names = NULL)
```

Arguments

`MCMC.output` The standard MCMC output file generated from a BEDASSLE run.

`percent.burnin` The percent of the sampled MCMC generations to be discarded as "burn-in." If the MCMC is run for 1,000,000 generations, and sampled every 1,000 generations, there will be 1,000 sampled generations. A `percent.burnin` of `20` will discard the first 200 sampled parameter values from that sample.

`thinning` The multiple by which the sampled MCMC generations are thinned. A `thinning` of `5` will sample every 5th MCMC generation.

`population.names`
: A vector of length `k`, where `k` is the number of populations/individuals (i.e. `k = nrow(counts)`), giving the name or identifier of each population/individual included in the analysis. These will be used to title the `k` trace plots of the phi

parameters estimated for each population/individual in the beta-binomial model. If the binomial model is used, `population.names` will not be used by this function.

Details

A trace plot is a basic visual tool for assessing MCMC mixing. If the chain is mixing well, the trace plot will resemble a "fuzzy caterpillar." If the trace plot has not plateaued, it is an indication that the chain has not converged on the stationary posterior distribution, and must be run longer. If the trace plot of a parameter exhibits high autocorrelation, the user may wish to either increase or decrease the scale of the tuning parameter on that parameter, to decrease or increase acceptance rates, respectively. If the chain appears to be bouncing between areas of "fuzzy caterpillar-dom," it may be an indication of a multi-modal likelihood surface.

Author(s)

Gideon Bradburd

`plot_joint_marginal` *Plots the joint marginal for a pair of parameters*

Description

For each sampled MCMC generation, the values estimated for a pair of parameters are logged and plotted against one another. Points are color coded by when in the analysis they were sampled, so that users can visually assess mixing.

Usage

```
plot_joint_marginal(parameter1, parameter2, percent.burnin = 0, thinning = 1,
param.name1 = deparse(substitute(parameter1)),
param.name2 = deparse(substitute(parameter2)))
```

Arguments

`parameter1`	One of the two parameters for which the joint marginal is being plotted.
`parameter2`	The other of the two parameters for which the joint marginal is being plotted.
`percent.burnin`	The percent of the sampled MCMC generations to be discarded as "burn-in." If the MCMC is run for 1,000,000 generations, and sampled every 1,000 generations, there will be 1,000 sampled generations. A `percent.burnin` of `20` will discard the first 200 sampled parameter values from that sample.
`thinning`	The multiple by which the sampled MCMC generations are thinned. A `thinning` of `5` will sample every 5th MCMC generation.
`param.name1`	The name of one of the two parameters for which the joint marginal is being plotted.
`param.name2`	The name of the other of the two parameters for which the joint marginal is being plotted.

Details

Visualizations of the joint marginal distribution allow users to (1) assess how well the MCMC is mixing, and (2) potentially diagnose instances of non-identifiability in the model. Strong linear trends in the joint marginal, or visible "ridges" in the likelihood surface, may be indicative of parameter non-identifiability, in which multiple combinations of values of these two parameters provide equally reasonable fits to the data.

Author(s)

Gideon Bradburd

`plot_marginal` *Plots the marginal density of a parameter*

Description

Plots the posterior marginal density of a parameter. Users may specify whether they want a histogram, a density, or both.

Usage

```
plot_marginal(parameter, percent.burnin = 0, thinning = 1, histogram = TRUE,
density = TRUE, population.names = NULL, param.name = deparse(substitute(parameter)))
```

Arguments

`parameter`	The parameter for which the marginal plot is being generated.
`percent.burnin`	The percent of the sampled MCMC generations to be discarded as "burn-in." If the MCMC is run for 1,000,000 generations, and sampled every 1,000 generations, there will be 1,000 sampled generations. A `percent.burnin` of `20` will discard the first 200 sampled parameter values from that sample.
`thinning`	The multiple by which the sampled MCMC generations are thinned. A `thinning` of `5` will sample every 5th MCMC generation.
`histogram`	A switch that controls whether or not the plot contains a histogram of the values estimated for the parameter over the course of the MCMC. Default is `TRUE`.
`density`	A switch that controls whether or not the plot shows the density of the values estimated for the parameter over the course of the MCMC. Default is `TRUE`.
`population.names`	A vector of length `k`, where `k` is the number of populations/individuals (i.e. `k = nrow(counts)`), giving the name or identifier of each population/individual included in the analysis. These will be used to title the `k` marginal plots of the phi parameters estimated for each population/individual in the beta-binomial model. If the binomial model is used, `population.names` will not be used by this function.
`param.name`	The name of the parameter for which the trace plot is being displayed.

Details

The marginal plot is another basic visual tool for MCMC diagnosis. Users should look for marginal plots that are "smooth as eggs" (indicating that the chain has been run long enough) and unimodal (indicating a single peak in the likelihood surface).

Author(s)

Gideon Bradburd

`plot_phi_marginal` *Plots the marginal for the phi parameter estimated in a single population*

Description

Plots the posterior marginal density of a phi parameter. Users may specify whether they want a histogram, a density, or both. For convenience, the F_k statistic is presented in place of the phi parameter, as this is the statistic users care about. F_k is defined as $\frac{1}{1+phi_k}$.

Usage

```
plot_phi_marginal(phi, percent.burnin = 0, thinning = 1, population.names = NULL,
pop.index = NULL,histogram = TRUE, density = TRUE)
```

Arguments

`phi` The vector of phi values estimated for a single population from an MCMC run.

`percent.burnin` The percent of the sampled MCMC generations to be discarded as "burn-in." If the MCMC is run for 1,000,000 generations, and sampled every 1,000 generations, there will be 1,000 sampled generations. A `percent.burnin` of `20` will discard the first 200 sampled parameter values from that sample.

`thinning` The multiple by which the sampled MCMC generations are thinned. A `thinning` of `5` will sample every 5th MCMC generation.

`population.names` The name of the population/individual for which the marginal density of the phi parameter is being plotted. This will be used to title the marginal plot. If `population.names` is not provided (i.e. `population.names = NULL`), a population index number will be used to title the plot.

`pop.index` A population index number generated to title a marginal plot if no `population.names` is specified.

`histogram` A switch that controls whether or not the plot contains a histogram of the values estimated for the parameter over the course of the MCMC. Default is `TRUE`.

`density` A switch that controls whether or not the plot shows the density of the values estimated for the parameter over the course of the MCMC. Default is `TRUE`.

Details

The marginal plot is another basic visual tool for MCMC diagnosis. Users should look for marginal plots that are "smooth as eggs" (indicating that the chain has been run long enough) and unimodal (indicating a single peak in the likelihood surface).

Author(s)

Gideon Bradburd

`plot_phi_trace` *Plots the trace plot for the phi parameter estimated in a single population*

Description

This function plots the phi parameter value estimated in each sampled generation of the MCMC against the index of that sampled generation. For convenience, the F_k statistic is presented in place of the phi parameter, as this is the statistic users care about. F_k is defined as $\frac{1}{1+phi_k}$.

Usage

```
plot_phi_trace(phi, percent.burnin = 0, thinning = 1, population.names = NULL,
pop.index = NULL)
```

Arguments

`phi` The vector of phi values estimated for a single population from an MCMC run.

`percent.burnin` The percent of the sampled MCMC generations to be discarded as "burn-in." If the MCMC is run for 1,000,000 generations, and sampled every 1,000 generations, there will be 1,000 sampled generations. A `percent.burnin` of `20` will discard the first 200 sampled parameter values from that sample.

`thinning` The multiple by which the sampled MCMC generations are thinned. A `thinning` of `5` will sample every 5th MCMC generation.

`population.names` The name of the population/individual for which the marginal density of the phi parameter is being plotted. This will be used to title the marginal plot. If `population.names` is not provided (i.e. `population.names = NULL`), a population index number will be used to title the plot.

`pop.index` A population index number generated to title a marginal plot if no `population.names` is specified.

Details

A trace plot is a basic visual tool for assessing MCMC mixing. If the chain is mixing well, the trace plot will resemble a "fuzzy caterpillar." If the trace plot has not plateaued, it is an indication that the chain has not converged on the stationary posterior distribution, and must be run longer. If the trace plot of a parameter exhibits high autocorrelation, the user may wish to either increase or decrease the scale of the tuning parameter on that parameter, to decrease or increase acceptance rates, respectively. If the chain appears to be bouncing between areas of "fuzzy caterpillar-dom," it may be an indication of a multi-modal likelihood surface.

Author(s)

Gideon Bradburd

plot_posterior_predictive_samples

Plots posterior predictive sampling

Description

This function plots the posterior predictive samples generated by `posterior.predictive.sample` around the observed data, so that users can evaluate how well the model is able to describe their data.

Usage

```
plot_posterior_predictive_samples(posterior.predictive.sample.file, save.figure = NULL,
figure.name = NULL)
```

Arguments

`posterior.predictive.sample.file`
: The output file generated by the function `posterior.predictive.sample`.

`save.figure`
: If `save.figure = TRUE`, a .png file of the posterior predictive sample plot will be saved, rather than plotted to the default R plotting window. Depending on the `posterior.predictive.sample.size` specified in `posterior.predictive.sample`, this figure may be time-intensive to generate, so users may wish to save the figure for future analysis, rather than re-plot it. If `save.figure = TRUE`, a `figure.name` must be specified. Default is `save.figure = NULL`.

`figure.name`
: If `save.figure = TRUE`, a .png file of the posterior predictive sample plot will be saved under the name `figure.name`. If `save.figure = TRUE`, a `figure.name` must be specified. Default is `figure.name = NULL`.

Details

This function plots posterior predictive unbiased pairwise F_{ST} around the observed unbiased pairwise F_{ST} values to determine how well the model is able to describe the user's data. Users should examine these plots to make sure that the model is picking up general trends (e.g. the slopes of isolation by geographic distance and isolation by ecological distance), and also to identify specific populations whose relationships with their neighbors are being poorly accommodated by the model.

Author(s)

Gideon Bradburd

plot_trace — *Plot the trace plot for a parameter*

Description

This function plots the parameter value estimated in each sampled generation of the MCMC against the index of that sampled generation.

Usage

```
plot_trace(parameter, percent.burnin = 0, thinning = 1,
param.name = deparse(substitute(parameter)))
```

Arguments

`parameter`	The parameter for which the trace plot is being generated.
`percent.burnin`	The percent of the sampled MCMC generations to be discarded as "burn-in." If the MCMC is run for 1,000,000 generations, and sampled every 1,000 generations, there will be 1,000 sampled generations. A `percent.burnin` of `20` will discard the first 200 sampled parameter values from that sample.
`thinning`	The multiple by which the sampled MCMC generations are thinned. A `thinning` of `5` will sample every 5th MCMC generation.
`param.name`	The name of the parameter for which the trace plot is being displayed.

Details

A trace plot is a basic visual tool for assessing MCMC mixing. If the chain is mixing well, the trace plot will resemble a "fuzzy caterpillar." If the trace plot has not plateaued, it is an indication that the chain has not converged on the stationary posterior distribution, and must be run longer. If the trace plot of a parameter exhibits high autocorrelation, the user may wish to either increase or decrease the scale of the tuning parameter on that parameter, to decrease or increase acceptance rates, respectively. If the chain appears to be bouncing between areas of "fuzzy caterpillar-dom," it may be an indication of a multi-modal likelihood surface.

Author(s)

Gideon Bradburd

posterior.predictive.sample

Generates posterior predictive samples

Description

This function simulates data using the inference model parameterized from the joint posterior of the MCMC and the observed independent variables ($D_{ij} and E_{ij}$). These posterior predictive samples can be compared to the observed data to see how well the model is able to describe the observed data.

Usage

```
posterior.predictive.sample(MCMC.output, posterior.predictive.sample.size, output.file,
prefix = "")
```

Arguments

`MCMC.output` The standard MCMC output file generated from a BEDASSLE run.

`posterior.predictive.sample.size`
The number of posterior predictive datasets the user wishes to simulate.

`output.file` The name that will be assigned to the R object containing the posterior predictive datasets. The suffix ".Robj" will be appended to the user-specified name.

`prefix` If specified, this prefix will be added to the output file name.

Details

This function simulates datasets like those the user analyzed with BEDASSLE, using the same independent variables (`sample.sizes`, D_{ij} and E_{ij}) as in the user's dataset and plugging them into the inference model, which is parameterized by randomly drawing parameter values from the joint posterior output of the MCMC analysis. These posterior predictive simulated allelic count data are summarized as unbiased pairwise F_{ST} (using `calculate.all.pairwise.Fst`), which may then be compared to the observed unbiased pairwise F_{ST} to determine how well the model is able to describe the user's data. The output of `posterior.predictive.sample` can be visualized using `plot.posterior.predictive.sample`.

Author(s)

Gideon Bradburd

Index