Package 'poolABC'

July 23, 2025

Title Approximate Bayesian Computation with Pooled Sequencing Data

Version 1.0.0

Description Provides functions to simulate Pool-seq data under models of demographic formation and to import Pool-seq data from real populations. Implements two ABC algorithms for performing parameter estimation and model selection using Pool-seq data. Cross-validation can also be performed to assess the accuracy of ABC estimates and model choice. Carvalho et al., (2022) <doi:10.1111/1755-0998.13834>.

License GPL (>= 3)

Encoding UTF-8

RoxygenNote 7.2.1

Imports doParallel, foreach, ggplot2, graphics, locfit, MetricsWeighted, nnet, poolHelper (>= 1.1.0), RColorBrewer, rlang, scrm, stats, utils

Depends R (>= 2.10)

LazyData true

URL https://github.com/joao-mcarvalho/poolABC

BugReports https://github.com/joao-mcarvalho/poolABC/issues

Suggests knitr, rmarkdown

VignetteBuilder knitr

NeedsCompilation no

Author João Carvalho [aut, cre, cph] (ORCID: <https://orcid.org/0000-0002-1728-0075>), Vítor Sousa [aut]

Maintainer João Carvalho <jgcarvalho@fc.ul.pt>

Repository CRAN

Date/Publication 2023-08-08 14:00:02 UTC

Contents

ABC . . . 3
cleanData . . . 6
cmd2pops . . . 7
cmdParallel . . . 8
cmdSingle . . . 10
createHeader . . . 11
createParams . . . 12
error_modelSel . . . 14
forceLocus . . . 15
getmode . . . 17
importContigs . . . 18
index.rejABC . . . 20
limits . . . 21
mergepost . . . 22
modelSelect . . . 24
mode_locfit . . . 26
multipleABC . . . 27
myparams . . . 29
params . . . 30
plot_errorABC . . . 30
plot_msel . . . 32
plot_param . . . 33
plot_Posteriors . . . 34
plot_stats . . . 36
plot_weighted . . . 37
poolSim . . . 39
poolStats . . . 45
poststat . . . 49
prepareData . . . 50
prepareFile . . . 52
priorsMatrix . . . 54
rc1 . . . 55
rc2 . . . 56
regABC . . . 57
rejABC . . . 58
remove_quantileReads . . . 60
remove_realReads . . . 61
runSCRM . . . 63
scaled.migration . . . 64
scaledPrior . . . 65
simulationABC . . . 66
sim_modelSel . . . 67
singleABC . . . 69
summary_modelSelect . . . 71
sumstats . . . 72

Index **73**

ABC *Parameter estimation with Approximate Bayesian Computation with several targets*

Description

Perform multivariate parameter estimation based on summary statistics using an Approximate Bayesian Computation (ABC) algorithm. This function always uses a rejection sampling algorithm while a local linear regression algorithm might or might not be used.

Usage

```
ABC(
  nPops,
  ntrials,
  freqs,
  positions,
  range,
  rMajor,
  rMinor,
  coverage,
  window,
  nLoci,
  limits,
  params,
  sumstats,
  tol,
  method,
  parallel = FALSE,
  ncores = NA
)
```

Arguments

nPops	is an integer indicating how many different populations are present in the dataset you are analysing.
ntrials	indicates how many different trials should be performed. Each trial corresponds to a different target for the parameter estimation.
freqs	is a list containing the allelic frequencies. Each entry of that list should represent a different contig and be a matrix where each row corresponds to a different site and each column to a different population.
positions	is a list containing the position of the SNPs. Each entry should represent a different contig and be a vector containing the position of each SNP present in the contig.

range	is a list containing the range of the contig. Each entry should represent a different contig and be a vector with two entries: the first detailing the minimum position of the contig and the second the maximum position of the contig.
rMajor	a list containing the number of major allele reads. Each entry should represent a different contig. For each contig (matrix), each row should be a different site and each column a different population.
rMinor	a list containing the number of minor allele reads. Each entry should represent a different contig. For each contig (matrix), each row should be a different site and each column a different population.
coverage	is a list containing the depth of coverage. Each entry should represent a different contig and be a matrix with the sites as rows and the different populations as columns.
window	is a non-negative integer indicating the size, in base pairs, of the block of the contig to keep.
nLoci	is a non-negative integer indicating how many different contigs should be kept in the output. If each randomly selected window is a different loci, then how many different window should be selected?
limits	is a matrix with two columns and as many rows as there are parameters. Each row should contain the minimum value of the prior for a given parameter in the first column and the maximum value in the second column.
params	is a vector or matrix of simulated parameter values i.e. numbers from the simulations. Each row or vector entry should be a different simulation and each column of a matrix should be a different parameter. This is the dependent variable for the regression, if a regression step is performed.
sumstats	is a vector or matrix of simulated summary statistics. Each row or vector entry should be a different simulation and each column of a matrix should be a different statistic. These act as the independent variables if a regression step is performed.
tol	is the tolerance rate, indicating the required proportion of points accepted nearest the target values.
method	either "rejection" or "regression" indicating whether a regression step should be performed during ABC parameter estimation.
parallel	logical, indicating whether this function should be run using parallel execution. The default setting is FALSE, meaning that this function will utilize a single core.
ncores	a non-negative integer that is required when parallel is TRUE. It specifies the number of cores to use for parallel execution.

Details

To use this function, the usual steps of ABC parameter estimation have to be performed. Briefly, data should have been simulated based on random draws from the prior distributions of the parameters of interest and a set of summary statistics should have been calculated from that data. This function requires as input the observed data and computes the same set of summary statistics from that observed data. Multiple sets of observed summary statistics are computed from ntrials sets

of nLoci blocks of size window. Parameter estimation is performed for each one of those sets of observed summary statistics i.e. each set corresponds to a different target.

After computing this set of observed summary statistics, a simple rejection is performed by calling the rejABC() function. In this step, parameter values are accepted if the Euclidean distance between the set of summary statistics computed from the simulated data and the set of summary statistics computed from the observed data is sufficiently small. The percentage of accepted simulations is determined by tol.

When method is "regression", a local linear regression method is used to correct for the imperfect match between the summary statistics computed from the simulated data and the summary statistics computed from the observed data. The output of the rejABC() function is used as the input of the regABC() function to apply this correction. The parameter values accepted in the rejection step are weighted by a smooth function (kernel) of the distance between the simulated and observed summary statistics and corrected according to a linear transformation.

Value

a list with seven different entries.

target	observed summary statistics.
ss	set of accepted summary statistics from the simulations.
unadjusted	parameter estimates obtained with the rejection sampling.
adjusted	regression adjusted parameter values.
predmean	estimates of the posterior mean for each parameter.
weights	regression weights.
position	position of each SNP used for calculating the observed summary statistics.

See Also

For more details see the poolABC vignette: vignette("poolABC", package = "poolABC")

Examples

```
# Note that this example is limited to a few of the options available
# you should check the poolABC vignette for more details

# this creates a variable with the path for the toy example data
mypath <- system.file('extdata', package = 'poolABC')

# import data for two populations from all files
mydata <- importContigs(path = mypath, pops = c(8, 10))

# to perform parameter inference for two populations using the rejection method
# and with a tolerance of 0.01
myabc <- ABC(nPops = 2, ntrials = 10, freqs = mydata$freqs, positions = mydata$positions,
range = mydata$range, rMajor = mydata$rMajor, rMinor = mydata$rMinor, coverage = mydata$coverage,
window = 1000, nLoci = 4, limits, params, sumstats, tol = 0.01, method = "rejection")

# the previous will perform parameter inference for 10 different targets (ntrials = 100)
```

```
# each of those trials will be comprised of 4 loci, each with 1000 base pairs

# to perform parameter inference for two populations using the regression method
# and with a tolerance of 0.01
myabc <- ABC(nPops = 2, ntrials = 10, freqs = mydata$freqs, positions = mydata$positions,
range = mydata$range, rMajor = mydata$rMajor, rMinor = mydata$rMinor, coverage = mydata$coverage,
window = 1000, nLoci = 4, limits, params, sumstats, tol = 0.01, method = "regression")
```

`cleanData` *Import and clean a single file containing data in* `popoolation2` *format*

Description

Imports data for two or four populations from a single file containing data in the _rc format. The data is then split so that the number of major-allele reads, minor-allele reads, total depth of coverage and remaining relevant information are kept on separate matrices.

Usage

```
cleanData(file, pops, header = NA, remove = NA, min.minor = NA)
```

Arguments

`file`	is a character string indicating the path to the file you wish to import.
`pops`	is a vector with the index of the populations that should be imported. This function works for two or four populations and so this vector must have either length 2 or 4.
`header`	is a character vector containing the names for the columns. If set to NA (default), no column names will be added to the output.
`remove`	is a character vector where each entry is a name of a contig to be removed. These contigs are, obviously, removed from the imported dataset. If NA (default), all contigs will be kept in the output.
`min.minor`	what is the minimum allowed number of reads with the minor allele across all populations? Sites where this threshold is not met are removed from the data. The default (NA) means that no sites will be removed because of their number of minor-allele reads.

Details

The information in the _rc format is stored in a x/y format, where x represents the observed reads and the y is the coverage. The initial step of this function splits this string to separate the number of reads from the total coverage. Then, the number of major plus minor allele reads is compared to the total coverage and sites where both values are not equal are removed from the dataset. Additionally, sites where any of the populations has an "N" as the reference character of their major allele, are removed from the data. This function also ensures that the major allele is the same and the most

frequent across all populations. Finally, if the `min.minor` input is supplied, sites where the total number of minor-allele reads is below the specified number, will be removed from the data set.

Note also that all non biallelic sites and sites where the sum of deletions in all populations is not zero will be removed from the dataset. Although this function can only import 2 or 4 populations at the time, it is possible to define which two or four populations to import. For instance, if we define the first population as the first column for which we have data in the x/y format, then you could wish to import the data for the 5th and 6th populations, defined as the populations in the 6th and 7th columns. To do so, you should define the `pops` input as `pops = c(5, 6)`.

Value

a list with the following elements:

`rMajor`	a matrix with the number of major-allele reads. Each row of this matrix is a different site and each column a different population.
`rMinor`	a matrix with the number of minor-allele reads. Each row of this matrix is a different site and each column a different population.
`coverage`	a matrix with the total coverage. Each row of this matrix is a different site and each column a different population.
`info`	a data frame with 5 different columns containing: the contig name, the SNP position, the reference character of the SNP and the reference character of the major and minor allele for each of the populations. Each row of this data frame corresponds to a different site

Examples

```
# load the data from one rc file
data(rc1)
# clean and organize the data in this single file
cleanData(file = rc1, pops = 7:10)
```

`cmd2pops` *Create SCRM command line for a model with two populations*

Description

This function creates a command line tailored for an isolation with migration model with two populations. The command line can then be fed to the scrm package to run the model.

Usage

```
cmd2pops(parameters, nSites, nLoci, nDip, mutrate, extra = FALSE)
```

Arguments

parameters	A vector where each entry corresponds to a different parameter, e.g. one entry is the size of the reference population, another is the time of recent split, etc. Please note that this functions depends on the ordering of the parameters in the vector and thus, it should only be used with a vector created with the createParams function.
nSites	An integer representing the number of base pairs that each locus should have.
nLoci	An integer that represents how many independent loci should be simulated.
nDip	An integer representing the total number of diploid individuals to simulate. Note that scrm actually simulates haplotypes, so the number of simulated haplotypes is double of this. Also note that this is the total number of diploid individuals and this function will distribute the individuals equally by the two populations.
mutrate	A number representing the mutation rate assumed for the simulations.
extra	is a logical value indicating whether the required number of loci should be enforced. The default is FALSE but, if set to TRUE, then additional loci will be simulated. These additional loci are simulated to try to have sufficient loci to keep the required number of loci after filtering.

Value

a character vector with two entries. The first entry is the scrm command line for the loci without any barriers against migration, while the second entry is the scrm command line for the loci without migration between divergent ecotypes.

Examples

```
# create a vector with parameter values for a two populations model
params <- createParams(Nref = c(25000, 25000), ratio = c(0.1, 3), pool = c(5, 250),
seq = c(0.0001, 0.001), split = c(0, 3), CW = c(1e-13, 1e-3), WC = c(1e-13, 1e-3),
bT = c(0, 0.2), model = "2pops")

# create the command line for the scrm package
cmd2pops(parameters = params, nSites = 2000, nLoci = 100, nDip = 100, mutrate = 2e-8)
```

---

cmdParallel *Create SCRM command line for a parallel origin scenario*

Description

This function creates a command line tailored for a scenario of parallel origin to explain ecotype formation. The command line can then be fed to the scrm package to run the model.

Usage

```
cmdParallel(parameters, nSites, nLoci, nDip, mutrate, extra = FALSE)
```

Arguments

parameters	A vector where each entry corresponds to a different parameter, e.g. one entry is the size of the reference population, another is the time of recent split, etc. Please note that this functions depends on the ordering of the parameters in the vector and thus, it should only be used with a vector created with the createParams function.
nSites	An integer representing the number of base pairs that each locus should have.
nLoci	An integer that represents how many independent loci should be simulated.
nDip	An integer representing the total number of diploid individuals to simulate. Note that scrm actually simulates haplotypes, so the number of simulated haplotypes is double of this. Also note that this is the total number of diploid individuals and this function will distribute the individuals equally by the two populations.
mutrate	A number representing the mutation rate assumed for the simulations.
extra	is a logical value indicating whether the required number of loci should be enforced. The default is FALSE but, if set to TRUE, then additional loci will be simulated. These additional loci are simulated to try to have sufficient loci to keep the required number of loci after filtering.

Details

For convenience, imagine we have two divergent ecotypes, named C and W. This model assumes that the first population corresponds to the C ecotype at the first location, the second population to the W ecotype in the first location, the third population to the C ecotype in the second location and the fourth population to the W ecotype in the second location.

Value

a character vector with four entries. The first entry is the scrm command line for the loci without any barriers against migration. The second entry is the command line for the loci without migration from the C towards the W ecotype. The third entry is command line for the loci without migration from the W towards the C ecotype and the last entry is the scrm command line for the loci without migration between divergent ecotypes.

Examples

```
# create a vector with parameter values for the parallel origin scenario
params <- createParams(Nref = c(25000, 25000), ratio = c(0.1, 3), pool = c(5, 250),
seq = c(0.0001, 0.001), split = c(0, 3), CW = c(1e-13, 1e-3), WC = c(1e-13, 1e-3),
CC =  c(1e-13, 1e-3), WW = c(1e-13, 1e-3), ANC = c(1e-13, 1e-3), bT = c(0, 0.2),
bCW = c(0, 0.5), bWC = c(0, 0.5), model = "Parallel")

# create the command line for the scrm package
cmdParallel(parameters = params, nSites = 2000, nLoci = 100, nDip = 400, mutrate = 2-8)
```

`cmdSingle` *Create SCRM command line for a single origin scenario*

Description

This function creates a command line tailored for a scenario of single origin to explain ecotype formation. The command line can then be fed to the scrm package to run the model.

Usage

```
cmdSingle(parameters, nSites, nLoci, nDip, mutrate, extra = FALSE)
```

Arguments

`parameters`	A vector where each entry corresponds to a different parameter, e.g. one entry is the size of the reference population, another is the time of recent split, etc. Please note that this functions depends on the ordering of the parameters in the vector and thus, it should only be used with a vector created with the `createParams` function.
`nSites`	An integer representing the number of base pairs that each locus should have.
`nLoci`	An integer that represents how many independent loci should be simulated.
`nDip`	An integer representing the total number of diploid individuals to simulate. Note that scrm actually simulates haplotypes, so the number of simulated haplotypes is double of this. Also note that this is the total number of diploid individuals and this function will distribute the individuals equally by the two populations.
`mutrate`	A number representing the mutation rate assumed for the simulations.
`extra`	is a logical value indicating whether the required number of loci should be enforced. The default is FALSE but, if set to TRUE, then additional loci will be simulated. These additional loci are simulated to try to have sufficient loci to keep the required number of loci after filtering.

Details

For convenience, imagine we have two divergent ecotypes, named C and W. This model assumes that the first population corresponds to the C ecotype at the first location, the second population to the C ecotype in the second location, the third population to the W ecotype in the first location and the fourth population to the W ecotype in the second location.

Value

a character vector with four entries. The first entry is the scrm command line for the loci without any barriers against migration. The second entry is the command line for the loci without migration from the C towards the W ecotype. The third entry is command line for the loci without migration from the W towards the C ecotype and the last entry is the scrm command line for the loci without migration between divergent ecotypes.

Examples

```
# create a vector with parameter values for the single origin scenario
params <- createParams(Nref = c(25000, 25000), ratio = c(0.1, 3), pool = c(5, 250),
seq = c(0.0001, 0.001), split = c(0, 3), CW = c(1e-13, 1e-3), WC = c(1e-13, 1e-3),
CC =  c(1e-13, 1e-3), WW = c(1e-13, 1e-3), ANC = c(1e-13, 1e-3), bT = c(0, 0.2),
bCW = c(0, 0.5), bWC = c(0, 0.5), model = "Single")

# create the command line for the scrm package
cmdSingle(parameters = params, nSites = 2000, nLoci = 100, nDip = 400, mutrate = 2-8)
```

---

createHeader *Create a header for a _rc file of popoolation2*

Description

Creates a header for files in the _rc format of the popoolation2 software. This header can be applied to a matrix as column names.

Usage

```
createHeader(nPops)
```

Arguments

nPops	is an integer specifying how many different populations exist in the _rc file.

Details

Please note that the first 9 columns are a default output of the popoolation2 software and thus this functions maintains the same names.

Value

a character vector with the column names for a _rc popoolation2 file.

Examples

```
createHeader(nPops = 10)
```

createParams *Draw parameters from the priors*

Description

This function creates a named vector of parameters that can be used as input in the command line of the scrm package. Please note that this function needs to be adjusted if you wish to test the effect of different prior distributions.

Usage

```
createParams(
  Nref,
  ratio,
  split,
  pool,
  seq,
  CW,
  WC,
  CC = NA,
  WW = NA,
  ANC = NA,
  bT,
  bCW = NA,
  bWC = NA,
  model,
  digits = 5
)
```

Arguments

Nref	The minimum and maximum value of the uniform distribution for the effective population size of the reference population (Nref).
ratio	The minimum and maximum value of the distribution from which the relative size of the present-day and ancestral populations are drawn. The size of these populations is set as a ratio of the size of the Nref population. All of these ratios are drawn from a log10 uniform distribution.
split	The minimum and maximum values, at the 4Nref scale, of the uniform distribution from which the values of the times of the split events are draw. Both the time of the recent split event and the distance between the two split events are drawn from this distribution.
pool	The minimum and maximum values of the uniform distribution from which the value of the error associated with DNA pooling is drawn. More specifically, this value is related with the unequal individual contribution to the pool.

`seq` The minimum and maximum values of the uniform distribution from which the value of the error associated with DNA sequencing is drawn. This parameter should be supplied as a decimal number between zero and one.

`CW` The minimum and maximum value of the uniform distribution from which the migration rate between the two divergent ecotypes inhabiting the same location is drawn. We consider that this parameter is drawn on a m scale. This is the migration rate from ecotype C to ecotype W.

`WC` The minimum and maximum value of the uniform distribution from which the migration rate between the two divergent ecotypes inhabiting the same location is drawn. We consider that this parameter is drawn on a m scale. This is the migration rate from ecotype W to ecotype C.

`CC` The minimum and maximum value of the uniform distribution from which the migration rate between similar ecotypes inhabiting different locations is drawn. We consider that this parameter is drawn on a m scale. This is the migration between the two C ecotypes at two different locations.

`WW` The minimum and maximum value of the uniform distribution from which the migration rate between similar ecotypes inhabiting different locations is drawn. We consider that this parameter is drawn on a m scale. This is the migration between the two W ecotypes at two different locations.

`ANC` The minimum and maximum value of the uniform distribution from which the migration rate between the two ancestral populations is drawn. We consider that this parameter is drawn on a m scale.

`bT` The minimum and maximum values of the distribution from which the proportion of the simulated loci where no migration occurs between divergent ecotypes is drawn. The maximum value should not be higher than one.

`bCW` The minimum and maximum values of the distribution from which the proportion of the simulated loci where no migration occurs from the C ecotype towards the W ecotype is drawn. The maximum value should not be higher than one.

`bWC` The minimum and maximum values of the distribution from which the proportion of the simulated loci where no migration occurs from the W ecotype towards the C ecotype is drawn. The maximum value should not be higher than one.

`model` Either "2pops", "Single" or "Parallel" indicating for which model should parameters be drawn.

`digits` An optional integer indicating the number of decimal places to use when rounding certain parameters. The default is five.

Value

a vector with one named entry per relevant parameter. Each entry is the sampled value from the prior for that particular parameter.

Examples

```
# for a model with two populations
createParams(Nref = c(25000, 25000), ratio = c(0.1, 3), pool = c(5, 250), seq = c(0.0001, 0.001),
split = c(0, 3), CW = c(1e-13, 1e-3), WC = c(1e-13, 1e-3), bT = c(0, 0.2), model = "2pops")
```

```
# for a single origin scenario
createParams(Nref = c(25000, 25000), ratio = c(0.1, 3), pool = c(5, 250), seq = c(0.0001, 0.001),
split = c(0, 3), CW = c(1e-13, 1e-3), WC = c(1e-13, 1e-3), CC =  c(1e-13, 1e-3),
WW = c(1e-13, 1e-3), ANC = c(1e-13, 1e-3), bT = c(0, 0.2), bCW = c(0, 0.5),
bWC = c(0, 0.5), model = "Single")
```

`error_modelSel` *Compute error in model selection with Approximate Bayesian Computation*

Description

This function calculates the confusion matrix and the mean misclassification probabilities of models from the output of the `sim_modelSel()` function.

Usage

```
error_modelSel(object, threshold = NA, print = TRUE)
```

Arguments

`object`	a list created by the `sim_modelSel()` function, containing results of a simulation study to evaluate the quality of model selection with Approximate Bayesian Computation.
`threshold`	numeric value between 0 and 1 representing the minimum posterior probability of assignment.
`print`	logical, if TRUE (default), then this function prints the mean models probabilities.

Details

It is also possible to define a `threshold` for the posterior model probabilities. This threshold sets the minimum posterior probability of assignment. Thus, a simulation where the posterior probability of any model is below the threshold will not be assigned to a model and will instead be classified as "unclear".

Value

apart from directly displaying the results if print is TRUE, the output object of this function is a list with the following elements:

`confusion.matrix`	the confusion matrix.
`probs`	the mean model misclassification probabilities.
`postmeans`	the mean model misclassification probabilities when each model is correctly or incorrectly estimated.

Examples

```
# load the matrix with simulated parameter values
data(sumstats)

# select a random simulation to act as target just to test the function
target <- sumstats[10 ,]

# create a "fake" vector of model indices
# this assumes that half the simulations were from one model and the other half from other model
# this is not true but serves as an example of how to use this function
index <- c(rep("model1", nrow(sumstats)/2), rep("model2", nrow(sumstats)/2))

# perform a leave-one-out cross validation of model selection
mysim <- sim_modelSel(index = index, sumstats = sumstats, nval = 10, tol = 0.1)

# compute the confusion matrix and the mean misclassification probabilities
error_modelSel(object = mysim)
```

---

forceLocus *Force the simulations to contain the required number of loci*

Description

This function attempts to force the required number of loci after the filtering steps are performed.

Usage

```
forceLocus(
  model,
  parameters,
  nSites,
  nLoci,
  nDip,
  mutrate,
  mean,
  variance,
  minimum,
  maximum,
  size,
  min.minor
)
```

Arguments

model	a character, either 2pops", "Single" or "Parallel" indicating which model should be simulated.

`parameters` a vector of parameters used to create the command line for the scrm package. Each entry of the vector is a different parameter. Note that each vector entry should be named with the name of the corresponding parameter. The output of the `CreateParameters` function is the intended input.

`nSites` is an integer that specifies how many base pairs should scrm simulate, i.e. how many sites per locus to simulate.

`nLoci` an integer that represents how many independent loci should be simulated.

`nDip` an integer representing the total number of diploid individuals to simulate. Note that scrm actually simulates haplotypes, so the number of simulated haplotypes is double of this. Also note that this is the total number of diploid individuals and this function will distribute the individuals equally by the simulated populations.

`mutrate` an integer representing the mutation rate assumed for the simulations.

`mean` an integer or a vector defining the mean value of the negative binomial distribution from which different number of reads are drawn. It represents the mean coverage across all sites. If a vector is supplied, the function assumes that each entry of the vector is the mean for a different population.

`variance` an integer or a vector defining the variance of the negative binomial distribution from which different number of reads are drawn. It represents the variance of the total coverage across all sites. If a vector is supplied, the function assumes that each entry of the vector is the variance for a different population.

`minimum` an integer representing the minimum coverage allowed. Sites where any population has a depth of coverage below this threshold are removed from the data.

`maximum` an integer representing the maximum coverage allowed. Sites where any population has a depth of coverage above this threshold are removed from the data.

`size` a list with one entry per population. Each entry should be a vector containing the size (in number of diploid individuals) of each pool. Thus, if a population was sequenced using a single pool, the vector should contain only one entry. If a population was sequenced using two pools, each with 10 individuals, this vector should contain two entries and both will be 10.

`min.minor` is an integer representing the minimum allowed number of minor-allele reads. Sites that, across all populations, have less minor-allele reads than this threshold will be removed from the data.

Details

This is done by simulating extra loci for each of the different types of simulations performed. The possible types of simulations include loci without barriers against migration between divergent ecotypes, loci without migration from the C towards the W ecotype, loci without migration from the W towards the C ecotypes and loci where no migration occurs between divergent ecotypes. Using this function, more loci than required are simulated for each of those types of simulations.

Then, a coverage-based filter is applied to the data, followed by a filter based on a required number of minor-allele reads per site. Those filters remove some loci from the data. The extra simulated loci should allow us to keep the required number of loci per type of simulation even after filtering.

Value

a list with two names entries

pool	a list with three different entries: major, minor and total. This list is obtained by running the forcePool function.
nPoly	a numeric value indicating the mean number of polymorphic sites across all simulated locus.

Examples

```
# create a vector with parameter values for a two populations model
params <- createParams(Nref = c(25000, 25000), ratio = c(0.1, 3), pool = c(5, 250),
seq = c(0.0001, 0.001), split = c(0, 3), CW = c(1e-13, 1e-3), WC = c(1e-13, 1e-3),
bT = c(0, 0.2), model = "2pops")

# simulate exactly 10 loci - using an isolation with migration model with two populations
forceLocus(model = "2pops", parameters = params, nSites = 1000, nLoci = 10, nDip = 100,
mutrate = 2e-8, mean = c(100, 100), variance = c(250, 250), minimum = 10, maximum = 200,
size = list(50, 50), min.minor = 0)
```

---

getmode *Calculate the mode of a distribution*

Description

Computes and outputs the mode of the input distribution.

Usage

```
getmode(x, xlim, weights = NULL, alpha = 0.7, precision = 1000)
```

Arguments

x	is a numeric vector containing the values of the distribution.
xlim	is a vector with two entries.The first entry is the minimum of the x distribution and the second entry is the maximum value of the x distribution. Ideally these values should be the minimum and maximum value of the prior for this particular parameter.
weights	this is an optional input consisting of a vector with the prior weights for the locfit function.
alpha	numeric value with the alpha parameter of the locfit function. The default value is 0.7
precision	value indicating the number of entries evaluated. The larger the value the higher the precision. The default value is 1000.

Details

The `locfit::locfit()` function is used to fit a local regression to the distribution. The `stats::predict()` function is then used to predict the y-axis values of the locfit and the mode is defined as the value where that prediction is maximized. Note that if this function is not able to fit a local regression to the distribution, then the mode of the distribution will be assumed to be equal to the median.

Value

a numeric value of the mode of the input distribution.

Examples

```
# create a random distribution
x <- rnorm(n = 100, mean = 2, sd = 25)

# compute the mode of the distribution
getmode(x = x, xlim = c(min(x), max(x)))
```

---

importContigs *Import multiple files containing data in PoPoolation2 format*

Description

Imports multiple files containing data in PoPoolation2 format and organize that information into different entries for each contig.

Usage

```
importContigs(
  path,
  pops,
  files = NA,
  header = NA,
  remove = NA,
  min.minor = NA,
  filter = FALSE,
  threshold = NA
)
```

Arguments

path	is a character string indicating the path to the folder where the data you wish to import is located.
pops	is a vector with the index of the populations that should be imported. This function works for two or four populations and so this vector must have either length 2 or 4.

files	is an integer or a numeric vector with the index of the files you wish to import.
header	is a character vector containing the names for the columns. If set to NA (default), no column names will be added to the output.
remove	is a character vector where each entry is a name of a contig to be removed. These contigs are, obviously, removed from the imported dataset. If NA (default), all contigs will be kept in the output.
min.minor	what is the minimum allowed number of reads with the minor allele across all populations? Sites where this threshold is not met are removed from the data.
filter	is a logical switch, either TRUE or FALSE. If TRUE, then the data is filtered by the frequency of the minor allele and if FALSE, that filter is not applied.
threshold	is the minimum allowed frequency for the minor allele. Sites where the allelic frequency is below this threshold are removed from the data.

Details

The data from two or four populations is split so that the number of major-allele reads, minor-allele reads, total depth of coverage and remaining relevant information are kept on separate list entries. Sites where the sum of the major and minor allele reads does not match the total coverage and sites where any population has an "N" as the reference character of their major allele, are removed from the data. This function also ensures that the major allele is the same and the most frequent across all populations. Note also that all non biallelic sites and sites where the sum of deletions in all populations is not zero will be removed from the dataset.

If the `min.minor` input is supplied, sites where the total number of minor-allele reads is below the specified number, will be removed from the data set. Alternatively, if the filter input is set to TRUE, data will be filtered by the frequency of the minor-allele. If a threshold is supplied, the computed frequency is compared to that threshold and sites where the frequency is below the threshold are removed from the dataset. If no threshold is supplied, the threshold is assumed to be `1/total coverage`, meaning that a site should have, at least, one minor-allele read.

Finally, the name of each contig is used to organize the information in a per contig basis. Thus, each output will be organized by contig. For example, the list with the number of minor-allele reads will contain several entries and each of those entries is a different contig.

Value

a list with six named entries:

freqs	a list with the allele frequencies, computed by dividing the number of minor-allele reads by the total coverage. Each entry of this list corresponds to a different contig. Each entry is a matrix where each row is a different site and each column is a different population.
positions	a list with the positions of each SNP. Each entry of this list is a vector corresponding to a different contig.
range	a list with the minimum and maximum SNP position of each contig. Each entry of this list is a vector corresponding to a different contig.
rMajor	a list with the number of major-allele reads. Each entry of this list corresponds to a different contig. Each entry is a matrix where each row is a different site and each column is a different population.

rMinor	a list with the number of minor-allele reads. Each entry of this list corresponds to a different contig. Each entry is a matrix where each row is a different site and each column is a different population.
coverage	a list with the total coverage. Each entry of this list corresponds to a different contig. Each entry is a matrix where each row is a different site and each column is a different population.

See Also

For more details see the poolABC vignette: `vignette("poolABC", package = "poolABC")`

Examples

```
# this function should be used to import your data
# you should include the path to the folder your PoPoolation2 data is

# this creates a variable with the path for the toy example data
mypath <- system.file('extdata', package = 'poolABC')

# an example of how to import data for two populations from all files
importContigs(path = mypath, pops = c(8, 10))

# to remove contigs from the data
importContigs(path = mypath, pops = c(8, 10), remove = "Contig1708")
```

---

index.rejABC *Parameter estimation with Approximate Bayesian Computation using rejection sampling and recording just the index of accepted simulations*

Description

This function performs multivariate parameter estimation based on summary statistics using an Approximate Bayesian Computation (ABC) algorithm. The algorithm used here is the rejection sampling algorithm. This is a simplified version of the `rejABC()` function that records only the index of the accepted simulations.

Usage

```
index.rejABC(target, params, sumstats, tol)
```

Arguments

target	a vector with the target summary statistics. These are usually the set of observed summary statistics.
params	is a vector or matrix of simulated parameter values i.e. numbers from the simulations. Each row or vector entry should be a different simulation and each column of a matrix should be a different parameter.

sumstats	is a vector or matrix of simulated summary statistics. Each row or vector entry should be a different simulation and each column of a matrix should be a different statistic.
tol	is the tolerance rate, indicating the required proportion of points accepted nearest the target values.

Details

The rejection sampling algorithm generates random samples from the posterior distributions of the parameters of interest. Note that to use this function, the usual steps of ABC parameter estimation have to be performed. Briefly, data should have been simulated based on random draws from the prior distributions of the parameters of interest and a set of summary statistics should have been calculated from that data. The same set of summary statistics should have been calculated from the observed data to be used as the `target` input in this function. Parameter values are accepted if the Euclidean distance between the set of summary statistics computed from the simulated data and the set of summary statistics computed from the observed data is sufficiently small. The percentage of accepted simulations is determined by `tol`.

Value

a list with two named entries

index	the index of the accepted simulations.
dst	euclidean distances in the region of interest.

Examples

```
# load the matrix with parameter values
data(params)
# load the matrix with simulated parameter values
data(sumstats)

# select a random simulation to act as target just to test the function
target <- sumstats[10 ,]

# Parameter estimation using rejection sampling
index.rejABC(target = target, params = params, sumstats = sumstats[-10, ], tol = 0.01)
```

---

limits *Matrix of prior limits*

Description

this imports a matrix with the limits of the prior distribution for each parameter. Each row of the matrix is a different parameter, indicated by the row name. The matrix contains two columns, the first being the minimum value of the distribution and the second being the maximum value.

Usage

```
limits
```

Format

a matrix with 8 rows and 2 columns. Each of the rows corresponds to a different parameter:

N1 relative size of the first population. This population corresponds to the C ecotype.

N2 relative size of the second population. This population corresponds to the W ecotype.

Split time, in 4Nref scale, of the split event that creates the two populations.

PoolError error associated with DNA pooling.

SeqError error associated with DNA sequencing.

pM proportion of the genome with no barriers against gene flow. This is the proportion of simulated loci where migration occurs in both directions between the divergent ecotypes.

mig_CW scaled migration rate between the two divergent ecotypes This is the migration rate from ecotype C to ecotype W.

mig_WC scaled migration rate between the two divergent ecotypes This is the migration rate from ecotype W to ecotype C.

Source

simulations performed

---

mergepost *Merge posterior distributions*

Description

After using the `multipleABC()` function to perform parameter estimation with Approximate Bayesian Computation for several targets, this function can be used to merge the different posterior distributions.

Usage

```
mergepost(target, global, post, a = 0.5, wtreg = NULL)
```

Arguments

`target`	a matrix or a list with target mean sumstat, where each entry corresponds to a vector of size n (n = number of summary statistics) with the summary statistics of each subset of loci.
`global`	numeric vector of size n with mean summary statistics across all loci.
`post`	list with sample of posterior obtained for each subset of loci. Each entry of the list is a matrix where each line corresponds to an accepted simulations (size S) and each column corresponds to a parameter.

a	numeric value with the alpha parameter of the locfit function.
wtreg	(optional) list with the weights of regression method. Each entry of the list is a numeric vector with weights for each accepted simulation (size S).

Details

The posterior density will be estimated after simply merging the posteriors computed from all target subset of loci and after weighting the posterior of each target by its distance to the overall summary statistic mean. In other words, each posterior will be weighted according to the distance between the mean summary statistics of the subset of loci for which that posterior was computed and the mean across all loci, giving more weight to sets of loci with a mean closer to the overall mean.

Additionally, if the regression weights are available, each accepted point will be weighted by its regression weight and by distance of its associated target. The combination of these weights will be used to merge the multiple posteriors. The weighted mean, median, mode and quantiles will be computed for each of these different posterior merging methods by using the `weighted_stats()` and `mode_locfit()` functions. Note that this function requires the package **locfit**.

Value

list of locfit objects with the density of the posterior for each parameter and of mean, mode and quantiles obtained using weighted quantiles. The list has the following elements:

merge	obtained by simply merging all the posteriors into a single one and fitting a local regression without any prior weighting.
merged_stat	posterior point estimates for the corresponding merging method, merge. This includes the median, mean, mode and various quantiles of the posterior.
weighted	each target was weighted by its distance to the global summary statistics mean, giving more weight to the target subset of loci with mean summary statistics closer to the mean across the genome.
weighted_stat	posterior point estimates for the corresponding merging method, weighted. This includes the median, mean, mode and various quantiles of the posterior.
merge_reg	each accepted point was weighted by its regression weight.
merge_reg_stat	posterior point estimates for the corresponding merging method, merge_reg. This includes the median, mean, mode and various quantiles of the posterior.
weighted_reg	each target was weighted according to its distance to the overall mean and each point was weighted by its regression weight.
weighted_reg_stat	posterior point estimates for the corresponding merging method, weighted_reg. This includes the median, mean, mode and various quantiles of the posterior.

Details about the output can be found at: https://aakinshin.net/posts/weighted-quantiles/ and https://www.rdocumentation.org/packages/reldist/versions/1.6-6/topics/wtd.quantile

Examples

```
# load the matrix with parameter values
data(params)
# load the matrix with simulated parameter values
data(sumstats)
# load the matrix with the prior limits
data(limits)

# select some random simulations to act as target just to test the function
targets <- sumstats[c(11:20) ,]
# we should remove those random simulation from the sumstats and params matrices
sumstats <- sumstats[-c(11:20), ]; params <- params[-c(11:20), ]

# parameter estimation for multiple targets
myabc <- multipleABC(targets = targets, params = params, sumstats = sumstats, limits = limits,
tol = 0.01, method = "regression")

# select a random simulation to act as the global value of the summary statistics
# ideally this should be computed from the entirety of the observed data
global <- sumstats[50, ]

# merge the posterior distributions obtained in the previous step
mergepost(target = targets, global = global, post = myabc$adjusted, wtreg = myabc$weights)
```

`modelSelect` *Perform model selection with Approximate Bayesian Computation*

Description

Estimates posterior model probabilities using Approximate Bayesian Computation (ABC).

Usage

```
modelSelect(target, index, sumstats, tol, method, warning = TRUE)
```

Arguments

`target`	is a vector with the target summary statistics. These are usually computed from observed data.
`index`	is a vector of model indices. This can be a a character vector of model names, repeated as many times as there are simulations for each model. This vector will be coerced to factor and it must have the same length as `nrow(sumstats)` to indicate which row of the `sumstats` matrix belongs to which model.
`sumstats`	is a vector or matrix containing the simulated summary statistics for all the models. Each row or vector entry should be a different simulation and each column of a matrix should be a different statistic. The order must be the same as the order of the models in the `index` vector.

tol	is a numerical value, indicating the required proportion of points nearest the target values (tolerance).
method	a character string, either "rejection" or "regression", indicating which algorithm should be used for model selection.
warning	logical, if TRUE (default) warnings produced while running this function, mainly related with accepting simulations for just one of the models, will be displayed.

Details

Prior to using this function, simulations must have been performed under, at least, two different models. When method is "rejection", the posterior posterior probability of a given model is approximated by the proportion of accepted simulations of that particular model. Note that this approximation is only valid if all models where, a priori, equally likely and if the number of simulations performed is the same for all models. When the method is set to "regression", a multinomial logistic regression is used to estimate the posterior model probabilities. This multinomial regression is implemented in the multinom function.

Value

a list with the following elements:

method	the method used for model selection.
indices	a vector of model indices in the accepted region. In other words, this vector contains the name of the accepted model for each accepted point.
pred	a vector of model probabilities.
ss	the summary statistics in the accepted region.
weights	vector of regression weights when method is regression.
nmodels	the number of a priori simulations performed for each model.

Examples

```
# load the matrix with simulated parameter values
data(sumstats)

# select a random simulation to act as target just to test the function
target <- sumstats[10 ,]

# create a "fake" vector of model indices
# this assumes that half the simulations were from one model and the other half from other model
# this is not true but serves as an example of how to use this function
index <- c(rep("model1", nrow(sumstats)/2), rep("model2", nrow(sumstats)/2))

# perform model selection with ABC
modelSelect(target = target, index = index, sumstats = sumstats, tol = 0.01, method = "regression")
```

`mode_locfit` *Compute mode of a locfit object*

Description

This function computes and outputs the the mode of a locfit object.

Usage

```
mode_locfit(locx, xlim, precision = 1000)
```

Arguments

`locx`	is a locfit object.
`xlim`	is a vector with two entries.The first entry is the minimum of the distribution and the second entry is the maximum value of the distribution.
`precision`	value indicating the number of entries evaluated. The larger the value the higher the precision. The default value is 1000.

Details

The `stats::predict()` function is used to predict the y-axis values of the locfit object and the mode is defined as the value where that prediction is maximized.

Value

a numeric value of the mode of the input locfit object.

Examples

```
# create a random distribution
x <- rnorm(n = 1000, mean = 2, sd = 25)

# perform a local regression
loc <- locfit::locfit(~x)

# compute the mode of the locfit object
mode_locfit(locx = loc, xlim = c(min(x), max(x)))
```

multipleABC *Parameter estimation with Approximate Bayesian Computation for multiple targets*

Description

Perform multivariate parameter estimation based on summary statistics using an Approximate Bayesian Computation (ABC) algorithm. This function always uses a rejection sampling algorithm while a local linear regression algorithm might or might not be used.

Usage

```
multipleABC(
  targets,
  params,
  sumstats,
  limits,
  tol,
  method,
  parallel = FALSE,
  ncores = NA
)
```

Arguments

targets	a matrix of observed summary statistics. Each row will be considered a different target for parameter estimation. Each column should be a different summary statistics and these statistics should correspond to the statistics in the sumstats input.
params	is a vector or matrix of simulated parameter values i.e. numbers from the simulations. Each row or vector entry should be a different simulation and each column of a matrix should be a different parameter. This is the dependent variable for the regression, if a regression step is performed.
sumstats	is a vector or matrix of simulated summary statistics. Each row or vector entry should be a different simulation and each column of a matrix should be a different statistic. These act as the independent variables if a regression step is performed.
limits	is a matrix with two columns and as many rows as there are parameters. Each row should contain the minimum value of the prior for a given parameter in the first column and the maximum value in the second column.
tol	is the tolerance rate, indicating the required proportion of points accepted nearest the target values.
method	either "rejection" or "regression" indicating whether a regression step should be performed during ABC parameter estimation.

parallel	logical, indicating whether this function should be run using parallel execution. The default setting is FALSE, meaning that this function will utilize a single core.
ncores	a non-negative integer that is required when parallel is TRUE. It specifies the number of cores to use for parallel execution.

Details

To use this function, the usual steps of ABC parameter estimation have to be performed. Briefly, data should have been simulated based on random draws from the prior distributions of the parameters of interest and a set of summary statistics should have been calculated from that data. The same set of summary statistics should have been calculated from the observed data to be used as the targets in this function. Parameter values are accepted if the Euclidean distance between the set of summary statistics computed from the simulated data and the set of summary statistics computed from the observed data is sufficiently small. The percentage of accepted simulations is determined by tol. This function performs a simple rejection by calling the rejABC() function.

When method is "regression", a local linear regression method is used to correct for the imperfect match between the summary statistics computed from the simulated data and the summary statistics computed from the observed data. The output of the rejABC() function is used as the input of the regABC() function to apply this correction. The parameter values accepted in the rejection step are weighted by a smooth function (kernel) of the distance between the simulated and observed summary statistics and corrected according to a linear transformation.

Please note that this functions performs parameter estimation for multiple targets. The targets should contain multiple rows and each row will be treated as an independent target for parameter estimation.

Value

the returned object is a list containing the following components:

target	parameter estimates obtained with the rejection sampling.
ss	set of accepted summary statistics from the simulations.
unadjusted	parameter estimates obtained with the rejection sampling.
adjusted	regression adjusted parameter values.
predmean	estimates of the posterior mean for each parameter.
weights	regression weights.

Examples

```
# load the matrix with parameter values
data(params)
# load the matrix with simulated parameter values
data(sumstats)
# load the matrix with the prior limits
data(limits)

# select some random simulations to act as target just to test the function
targets <- sumstats[c(11:20) ,]
```

```
# we should remove those random simulation from the sumstats and params matrices
sumstats <- sumstats[-c(11:20), ]; params <- params[-c(11:20), ]

# parameter estimation for multiple targets
multipleABC(targets = targets, params = params, sumstats = sumstats, limits = limits,
tol = 0.01, method = "regression")
```

`myparams` *Matrix of simulated parameter values*

Description

This data set contains a matrix of simulated parameter values. These parameter values were sampled from prior distributions and used to perform simulations under a isolation with migration model with two populations. Each row of this matrix corresponds to a different simulation.

Usage

```
myparams
```

Format

a matrix with 5000 rows and 8 columns:

N1 relative size of the first population. This population corresponds to the C ecotype.

N2 relative size of the second population. This population corresponds to the W ecotype.

Split time, in 4Nref scale, of the split event that creates the two populations.

PoolError error associated with DNA pooling.

SeqError error associated with DNA sequencing.

mCW migration rate between the two divergent ecotypes This is the migration rate from ecotype C to ecotype W.

mWC migration rate between the two divergent ecotypes This is the migration rate from ecotype W to ecotype C.

pM proportion of the genome with no barriers against gene flow. This is the proportion of simulated loci where migration occurs in both directions between the divergent ecotypes.

Source

simulations performed

params *Matrix of simulated parameter values*

Description

This data set contains a matrix of simulated parameter values. These parameter values were sampled from prior distributions and used to perform simulations under a isolation with migration model with two populations. Each row of this matrix corresponds to a different simulation.

Usage

```
params
```

Format

a matrix with 10000 rows and 8 columns:

N1 relative size of the first population. This population corresponds to the C ecotype.

N2 relative size of the second population. This population corresponds to the W ecotype.

Split time, in 4Nref scale, of the split event that creates the two populations.

PoolError error associated with DNA pooling.

SeqError error associated with DNA sequencing.

pM proportion of the genome with no barriers against gene flow. This is the proportion of simulated loci where migration occurs in both directions between the divergent ecotypes.

mig_CW scaled migration rate between the two divergent ecotypes This is the migration rate from ecotype C to ecotype W.

mig_WC scaled migration rate between the two divergent ecotypes This is the migration rate from ecotype W to ecotype C.

Source

simulations performed

plot_errorABC *Prediction error plots for ABC using a list*

Description

Plots the prediction error computed from a leave-one-out cross validation for ABC parameter inference. This function takes as input a list created when performing cross validation and allows the user to select which ABC algorithm and point estimate statistic to plot.

Usage

```
plot_errorABC(
  x,
  method,
  statistic,
  index,
  transformation = "none",
  main = NULL
)
```

Arguments

x	is a list produced by a leave-one-out cross validation of ABC. This list should contain the prediction errors computed using the rejection and/or regression algorithm. For each of those methods, the prediction error obtained using three different point estimates of the posterior should be included in this list.
method	a character that can be either 'rej' or 'reg' indicating whether you wish to plot the prediction error computed with a rejection or regression based ABC algorithm.
statistic	a character that can be 'mode', 'median' or 'mean' indicating if you wish to plot the prediction error obtained using the mode, median or mean as the point estimate of the posterior.
index	an integer indicating which parameter to look at. It corresponds to a column on a matrix. So, to plot the first parameter, corresponding to the first column, select 1. To plot the second parameter, select 2 and so on.
transformation	default is none. It can also be 'log' if you wish to transform both the true and estimated values using a log10 scale.
main	is an optional character input. It will be used as the title of the plot. If NULL (default), then a generic title will be used instead.

Details

These plots help in visualizing the quality of the estimation and the effect of the chosen tolerance level or point estimate statistic.

Value

a plot of the estimated value of the parameter (in the y-axis) versus the true parameter value (in the x-axis). A line marking the perfect correspondence between the true and estimated values is also plotted. Thus, the closer the points are to that line, the lower the prediction error is.

Examples

```
# load the matrix with parameter values
data(params)
# load the matrix with simulated parameter values
data(sumstats)
# load the matrix with the prior limits
data(limits)
```

```
# perform a leave-one-out cross validation for ABC
mysim <- simulationABC(params = params, sumstats = sumstats, limits, nval = 10,
tol = 0.1, method = "regression")

# plot the prediction error for a given parameter
plot_errorABC(x = mysim, method = "reg", statistic = "median", index = 1)
```

`plot_msel` *Plot model misclassification*

Description

Displays a barplot of the confusion matrix obtained with a leave-one-out cross validation for model selection.

Usage

```
plot_msel(object, color = TRUE)
```

Arguments

`object`	a list created by the `error_modelSel()` function, containing the results of a leave-one-out cross validation for model selection.
`color`	logical, if TRUE (default) then a colour version of the barplot will be produced, if FALSE then a grey scale version will be produced.

Details

The barplot shows the proportion of validation simulations classified to each of the models. This function can produce either a colour or a grey scale barplot. If the classification of models is perfect, meaning that the model probability of each model is one for the correct model, then each bar will have a single colour representing its corresponding model.

Value

a barplot of the proportion of simulations classified to any of the models. In other words, a barplot of the confusion matrix.

Examples

```
# load the matrix with simulated parameter values
data(sumstats)

# select a random simulation to act as target just to test the function
target <- sumstats[10 ,]

# create a "fake" vector of model indices
```

```
# this assumes that half the simulations were from one model and the other half from other model
# this is not true but serves as an example of how to use this function
index <- c(rep("model1", nrow(sumstats)/2), rep("model2", nrow(sumstats)/2))

# perform a leave-one-out cross validation of model selection
mysim <- sim_modelSel(index = index, sumstats = sumstats, nval = 10, tol = 0.1)

# compute the confusion matrix and the mean misclassification probabilities
myerror <- error_modelSel(object = mysim, print = FALSE)

# barplot of model misclassification
plot_msel(object = myerror)
```

---

plot_param *Plot the density estimation of a given parameter*

Description

Plots the density estimation of a single parameter for quick visualization of the quality of an ABC analysis.

Usage

```
plot_param(prior, posterior, limits, index, weights = NULL)
```

Arguments

prior	is a vector or matrix of simulated parameter values i.e. numbers from the simulations. Each row or vector entry should be a different simulation and each column of a matrix should be a different parameter. This corresponds to the prior distribution and it should contain all the simulated parameter values.
posterior	is either a list or a matrix with samples from the posterior distributions obtained for each target. If in list format, each entry should be a matrix where each row corresponds to a different accepted simulations and each column corresponds to a different parameter.
limits	is a matrix with two columns and as many rows as there are parameters. Each row should contain the minimum value of the prior for a given parameter in the first column and the maximum value in the second column.
index	is an non-negative integer indicating which parameter to plot. It corresponds to the desired column of a matrix in the posteriors input. So, to plot the first parameter, corresponding to the first column in the posteriors input select 1. To plot the second parameter, select 2 and so on.
weights	is an optional list input containing the weights from the local linear regression method. Each entry of the list should be a numeric vector with the weights for each accepted simulation.

Details

This function can be used for a quick visualization of the posterior distribution obtained for a single target with the `singleABC()` function. Alternatively, if parameter estimation was performed with the `multipleABC()` function, the multiple posterior distributions, each obtained for a different target, will be combined into a single matrix and all values will be considered samples from the same posterior distribution.

Value

a plot of the density estimation of a given parameter. This plot will include a title with the name of the parameter. It will also include the density of the prior distribution for that parameter.

Examples

```
# load the matrix with parameter values
data(params)
# load the matrix with simulated parameter values
data(sumstats)
# load the matrix with the prior limits
data(limits)

# select a random simulation to act as target just to test the function
target <- sumstats[15 ,]
# we should remove the random simulation from the sumstats and params matrices
sumstats <- sumstats[-15, ]; params <- params[-15, ]

# parameter estimation for a single target
myabc <- singleABC(target = target, params = params, sumstats = sumstats, limits = limits,
tol = 0.01, method = "regression")

# plot the density estimation of a given parameter
plot_param(prior = params, posterior = myabc$adjusted, limits = limits,
index = 6, weights = myabc$weights)

# note that this is just an example!
# we don't have enough simulations to obtain credible results
```

---

plot_Posteriors *Plot multiple posterior distributions*

Description

Plots, in the same plot, the density of multiple posterior distributions of a given parameter.

Usage

```
plot_Posteriors(posteriors, index, limits, weights = NULL)
```

Arguments

posteriors	is a list with samples from the posterior distributions obtained for each target. Each entry of the list is a matrix where each row corresponds to a different accepted simulations and each column corresponds to a different parameter.
index	an non-negative integer indicating which parameter to plot. It corresponds to the desired column of a matrix in the posteriors input. So, to plot the first parameter, corresponding to the first column in the posteriors input select 1. To plot the second parameter, select 2 and so on.
limits	is a matrix with two columns and as many rows as there are parameters. Each row should contain the minimum value of the prior for a given parameter in the first column and the maximum value in the second column.
weights	is an optional list input containing the weights from the local linear regression method. Each entry of the list should be a numeric vector with the weights for each accepted simulation.

Details

After using the multipleABC() or ABC() functions to perform parameter estimation with Approximate Bayesian Computation with several targets, this function can be used for a quick visualization of the quality of an ABC analysis. Multiple posterior distributions, each obtained for a different target, are plotted in the same plot, allowing for a visualization of the shape of the posteriors and a quick inspection of whether all the posteriors converge to the same estimate.

Value

a plot with multiple posterior distributions, each obtained for a different target, for the selected parameter.

Examples

```
# load the matrix with parameter values
data(params)
# load the matrix with simulated parameter values
data(sumstats)
# load the matrix with the prior limits
data(limits)

# select some random simulations to act as target just to test the function
targets <- sumstats[c(11:20) ,]
# we should remove those random simulation from the sumstats and params matrices
sumstats <- sumstats[-c(11:20), ]; params <- params[-c(11:20), ]

# parameter estimation for a single target
myabc <- multipleABC(targets = targets, params = params, sumstats = sumstats, limits = limits,
tol = 0.01, method = "regression")

# plot multiple posteriors
plot_Posteriors(posteriors = myabc$adjusted, index = 1, limits = limits, weights = myabc$weights)
```

```
# note that this is just an example!
# we don't have enough simulations to obtain credible results
```

`plot_stats` *Plot the fit of a summary statistic to the target*

Description

Plot the fit of a summary statistic to the target

Usage

```
plot_stats(sumstat, target, accepted, index = NA, colour = TRUE)
```

Arguments

`sumstat`	is a vector or matrix of simulated summary statistics. If this input is a vector, then each entry should correspond to a different simulation. If it is a matrix, then each row should be a different simulation and each column a different statistic. Note that this should be the entire set of simulated values.
`target`	is an integer or a numeric vector containing the target of the parameter inference. If a single integer, then this should be the target summary statistic corresponding to the input `sumstat` vector. If this input is a vector, then the order of the entries in the vector should be the same as the order of the columns of the `sumstat` matrix input. Either way, this input should contain the value of the summary statistics calculated from observed data.
`accepted`	is a vector or matrix of accepted summary statistics. If this input is a vector, then each entry should correspond to a different simulation. If it is a matrix, then each row should be a different simulation and each column a different statistic. Note that this should be summary statistics of the accepted simulations during parameter inference.
`index`	is an optional non-negative integer. This input is only required when the `sumstat` and `accepted` inputs are matrices. In that instance, it will indicate which summary statistic to plot. It corresponds to the desired column of the `sumstat` and `accepted` matrices and to the entry of the `target` vector.
`colour`	logical, indicating whether the plot should be a colour version (default) or a grayscale plot.

Value

a plot with the fit of the simulated summary statistics to the observed value. Both the density estimation of the entire simulated summary statistics and the accepted summary statistics are contrasted with the observed value.

Examples

```
# load the matrix with parameter values
data(params)
# load the matrix with simulated parameter values
data(sumstats)
# load the matrix with the prior limits
data(limits)

# select a random simulation to act as target just to test the function
target <- sumstats[10 ,]
# we should remove the random simulation from the sumstats and params matrices
sumstats <- sumstats[-10, ]; params <- params[-10, ]

# parameter estimation for a single target
myabc <- singleABC(target = target, params = params, sumstats = sumstats,
limits = limits, tol = 0.01, method = "regression")

# check the fit of a summary statistic to the target
plot_stats(sumstat = sumstats, target = target, accepted = myabc$ss, index = 5)

# note that we performed parameter estimation for a single target
# because this function will only work when using a matrix
```

---

plot_weighted *Plot the density estimation of a given parameter*

Description

Plots a locfit object obtained after parameter estimation with Approximate Bayesian Computation using the `multipleABC()` function and merging the multiple posteriors with the `mergepost()` function.

Usage

```
plot_weighted(
  prior,
  merged_posterior,
  index,
  limits,
  regWeights = TRUE,
  weighted = TRUE
)
```

Arguments

`prior`	is a vector or matrix of simulated parameter values i.e. numbers from the simulations. Each row or vector entry should be a different simulation and each

column of a matrix should be a different parameter. This corresponds to the prior distribution and it should contain all the simulated parameter values.

`merged_posterior`
: is a list obtained by the `mergepost()` function. The output of that function produces a list with the locfit of the various parameters. This function plots those locfits.

`index`
: is an non-negative integer indicating which parameter to plot. It corresponds to the desired entry of the `merged_posterior` list. So, to plot the first parameter, corresponding to the first entry in the `merged_posterior` input select 1. To plot the second parameter, select 2 and so on.

`limits`
: is a matrix with two columns and as many rows as there are parameters. Each row should contain the minimum value of the prior for a given parameter in the first column and the maximum value in the second column.

`regWeights`
: logical, indicating whether to plot the posterior density obtained from merging the multiple posteriors with or without the weights of the regression step. The default is TRUE.

`weighted`
: logical, indicating whether to plot the posterior density obtained from merging the multiple posteriors with or without weighting by the overall distance to the global mean. The default is TRUE.

Details

The `mergepost()` function includes different posterior merging methods and produces locfit objects for each parameter and method. It is possible to select which parameter to plot, with the `index` input, and whether to plot the density estimation after each accepted point was weighted by its regression weight and by distance of its associated target to the overall mean of the data. If `regWeights` is set to FALSE, the density estimation obtained without considering the regression weights will be plotted. If `weighted` is set to FALSE, the density estimation obtained without considering the distance between the mean summary statistics of the target and the mean across all loci.

Value

a plot of the density estimation of a given parameter. This plot will include a title with the name of the parameter. It will also include the density of the prior distribution for that parameter. The density estimation shown here is obtained after merging multiple posteriors for that parameter.

Examples

```
# load the matrix with parameter values
data(params)
# load the matrix with simulated parameter values
data(sumstats)
# load the matrix with the prior limits
data(limits)

# select some random simulations to act as target just to test the function
targets <- sumstats[c(11:20) ,]
# we should remove those random simulation from the sumstats and params matrices
sumstats <- sumstats[-c(11:20), ]; params <- params[-c(11:20), ]
```

```
# parameter estimation for multiple targets
myabc <- multipleABC(targets = targets, params = params, sumstats = sumstats, limits = limits,
tol = 0.01, method = "regression")

# select a random simulation to act as the global value of the summary statistics
# ideally this should be computed from the entirety of the observed data
global <- sumstats[50, ]

# merge the posterior distributions obtained in the previous step
mymerge <- mergepost(target = targets, global = global, post = myabc$adjusted,
wtreg = myabc$weights)

# plot the merged posterior distribution
plot_weighted(prior = params, merged_posterior = mymerge, index = 7, limits = limits)

# note that this is just an example!
# we don't have enough simulations to obtain credible results
```

---

poolSim *Simulation of Pooled DNA sequencing*

Description

This is a master function that goes to all the steps required to obtain summary statistics from pooled sequencing data.

Usage

```
poolSim(
  model,
  nDip,
  nPops,
  size,
  nLoci,
  nSites,
  mutrate,
  mean,
  variance,
  minimum,
  maximum,
  min.minor = NA,
  Nref,
  ratio,
  split,
  pool,
  seq,
```

```
  CW = NA,
  WC = NA,
  CC = NA,
  WW = NA,
  ANC = NA,
  bT = NA,
  bCW = NA,
  bWC = NA,
  force = FALSE
)
```

Arguments

model	a character, either 2pops", "Single" or "Parallel" indicating which model should be simulated.
nDip	an integer representing the total number of diploid individuals to simulate. Note that scrm actually simulates haplotypes, so the number of simulated haplotypes is double of this. Also note that this is the total number of diploid individuals and this function will distribute the individuals equally by the simulated populations.
nPops	An integer, representing the total number of populations of the simulated model.
size	a list with one entry per population. Each entry should be a vector containing the size (in number of diploid individuals) of each pool. Thus, if a population was sequenced using a single pool, the vector should contain only one entry. If a population was sequenced using two pools, each with 10 individuals, this vector should contain two entries and both will be 10.
nLoci	an integer that represents how many independent loci should be simulated.
nSites	is an integer that specifies how many base pairs should scrm simulate, i.e. how many sites per locus to simulate.
mutrate	an integer representing the mutation rate assumed for the simulations.
mean	an integer or a vector defining the mean value of the negative binomial distribution from which different number of reads are drawn. It represents the mean coverage across all sites. If a vector is supplied, the function assumes that each entry of the vector is the mean for a different population.
variance	an integer or a vector defining the variance of the negative binomial distribution from which different number of reads are drawn. It represents the variance of the total coverage across all sites. If a vector is supplied, the function assumes that each entry of the vector is the variance for a different population.
minimum	an integer representing the minimum coverage allowed. Sites where any population has a depth of coverage below this threshold are removed from the data.
maximum	an integer representing the maximum coverage allowed. Sites where any population has a depth of coverage above this threshold are removed from the data.
min.minor	is an integer representing the minimum allowed number of minor-allele reads. Sites that, across all populations, have less minor-allele reads than this threshold will be removed from the data.
Nref	is the minimum and maximum value of the uniform distribution for the effective population size of the reference population (Nref).

`ratio` is the minimum and maximum value of the distribution from which the relative size of the present-day and ancestral populations are drawn. The size of these populations is set as a ratio of the size of the Nref population. All of these ratios are drawn from a log10 uniform distribution.

`split` is the minimum and maximum values, at the 4Nref scale, of the uniform distribution from which the values of the times of the split events are draw. Both the time of the recent split event and the distance between the two split events are drawn from this distribution.

`pool` is the the minimum and maximum values of the uniform distribution from which the value of the error associated with DNA pooling is drawn. More specifically, this value is related with the unequal individual contribution to the pool. This parameter should be supplied as a decimal number between zero and one.

`seq` is the minimum and maximum values of the uniform distribution from which the value of the error associated with DNA sequencing is drawn. This parameter should be supplied as a decimal number between zero and one.

`CW` is the minimum and maximum value of the uniform distribution from which the migration rate between the two divergent ecotypes inhabiting the same location is drawn. We consider that this parameter is drawn on a m scale. This is the migration rate from ecotype C to ecotype W.

`WC` is the minimum and maximum value of the uniform distribution from which the migration rate between the two divergent ecotypes inhabiting the same location is drawn. We consider that this parameter is drawn on a m scale. This is the migration rate from ecotype W to ecotype C.

`CC` is the minimum and maximum value of the uniform distribution from which the migration rate between similar ecotypes inhabiting different locations is drawn. We consider that this parameter is drawn on a m scale. This is the migration between the two C ecotypes at two different locations.

`WW` is the minimum and maximum value of the uniform distribution from which the migration rate between similar ecotypes inhabiting different locations is drawn. We consider that this parameter is drawn on a m scale. This is the migration between the two W ecotypes at two different locations.

`ANC` is the minimum and maximum value of the uniform distribution from which the migration rate between similar ecotypes inhabiting different locations is drawn. We consider that this parameter is drawn on a m scale. This is the migration between the two W ecotypes at two different locations.

`bT` is the minimum and maximum values of the distribution from which the proportion of the simulated loci where no migration occurs between divergent ecotypes is drawn. The maximum value should not be higher than one.

`bCW` is the minimum and maximum values of the distribution from which the proportion of the simulated loci where no migration occurs from the C ecotype towards the W ecotype is drawn. The maximum value should not be higher than one.

`bWC` is the minimum and maximum values of the distribution from which the proportion of the simulated loci where no migration occurs from the W ecotype towards the C ecotype is drawn. The maximum value should not be higher than one.

`force` is a logical value indicating whether the required number of loci should be enforced. The default is FALSE but, if set to TRUE, then additional loci will be simulated. These additional loci are simulated to try to have sufficient loci to keep the required number of loci after filtering.

Details

Starts by creating a vector of parameters, with values drawn from the respective prior distributions. Then those parameter values are used to simulate genetic data under a coalescent approach. A series of steps is then followed to turn that genetic data into pooled sequencing data. Finally, a set of summary statistics is computed using the simulated pooled sequencing data.

Value

a list with several named entries. The number of entries depends of the chosen model.

`Nref` numeric, sampled value from the prior for the effective population size of the reference population.

`N1` numeric, sampled value from the prior for the relative size of the present-day populations. This is the relative size of the first population.

`N2` numeric, sampled value from the prior for the relative size of the present-day populations. This is the relative size of the second population.

`N3` numeric, sampled value from the prior for the relative size of the present-day populations. This is the relative size of the third population. This entry only exists when the selected model has four populations.

`N4` numeric, sampled value from the prior for the relative size of the present-day populations. This is the relative size of the fourth population. This entry only exists when the selected model has four populations.

`NA1` numeric, sampled value from the prior for the relative size of the ancestral populations. This is the relative size of the ancestral population of N1 and N2. This entry only exists when the selected model has four populations.

`NA2` numeric, sampled value from the prior for the relative size of the ancestral populations. This is the relative size of the ancestral population of N3 and N4. This entry only exists when the selected model has four populations.

`Split` numeric, sampled value from the prior for the time, in 4Nref scale, of the recent split event.

`Dsplit` numeric, sampled value from the prior for the time, in 4Nref scale, of the distance between the two split events.

`PoolError` numeric, sampled value from the prior for the error associated with DNA pooling.

`SeqError` numeric, sampled value from the prior for the error associated with DNA sequencing.

`mCW1` numeric, sampled value from the prior for the migration rate between the two divergent ecotypes inhabiting the first location. This is the migration rate from ecotype C to ecotype W. For a two population model, this entry will be called mCW because that model considers a single location.

`mCW2` numeric, sampled value from the prior for the migration rate between the two divergent ecotypes inhabiting the second location. This is the migration rate from ecotype C to ecotype W. For a two population model, this entry will not exist.

`mWC1` numeric, sampled value from the prior for the migration rate between the two divergent ecotypes inhabiting the first location. This is the migration rate from ecotype W to ecotype C. For a two population model, this entry will be called mWC because that model considers a single location.

`mWC2` numeric, sampled value from the prior for the migration rate between the two divergent ecotypes inhabiting the second location. This is the migration rate from ecotype W to ecotype C. For a two population model, this entry will not exist.

`mCC` numeric, sampled value from the prior for the migration rate between similar ecotypes inhabiting different locations. This is the migration between the two C ecotypes at two different locations. For a two population model, this entry will not exist.

`mWW` numeric, sampled value from the prior for the migration rate between similar ecotypes inhabiting different locations. This is the migration between the two W ecotypes at two different locations. For a two population model, this entry will not exist.

`mAA` numeric, sampled value from the prior for the migration rate between the two ancestral populations. For a two population model, this entry will not exist.

`pM` numeric, sampled value from the prior for the proportion of the genome with no barriers against gene flow. This is the proportion of simulated loci where migration occurs in both directions between the divergent ecotypes.

`pCW` numeric, sampled value from the prior for the proportion of the genome where no migration occurs from the C ecotype towards the W ecotype. This is the proportion of simulated loci where migration occurs only from W towards C. This entry does not exist for the two populations model.

`pWC` numeric, sampled value from the prior for the proportion of the genome where no migration occurs from the W ecotype towards the C ecotype. This is the proportion of simulated loci where migration occurs only from C towards W. This entry does not exist for the two populations model.

`pNO` numeric, sampled value from the prior for the proportion of the genome with no gene flow between divergent ecotypes. This is the proportion of simulated loci where migration does not occur in both directions between the C and W ecotypes.

`nPoly` numeric, mean number of polymorphic sites across all simulated locus.

`nFilter` numeric, mean number of polymorphic sites retained after filtering across all simulated locus.

`nLoci` numeric, total number of loci retained after filtering. Summary statistics are calculated for these loci.

`Sf` numeric, fraction of sites fixed between populations. For the model with two populations, this is a single value. For the four-population models, this includes three values: the first is the fraction of fixed sites between the two populations

in the first location, the second value is between the populations in the second location and the third value is the overall fraction of fixed sites, obtained by comparing each population against the other three.

`Sx` numeric, fraction of exclusive sites per population. When running the model with two populations, this entry has two values - one per population. For the four-population models, there is also one value per population, followed by a fifth value representing the fraction of sites that are segregating in only one of the populations.

`SS` numeric values representing the fraction of sites shared between populations. For the model with two populations, this is a single value. When running one of the four-population models, this entry has three values. The first is the fraction of shared sites between the two populations in the first location, the second value is between the populations in the second location and the third value is the fraction of shared sites across all four populations.

`Mean_Het` numeric, expected heterozygosity within each population. This entry has two values when using a two populations model and four when running one of the four-populations model.

`SD_Het` numeric, standard deviation of the expected heterozygosity for each population. This entry has two values when using a two populations model and four when running one of the four-populations model.

`Mean_HetBet` numeric, mean heterozygosity between all pairs of populations. For the two populations model, this is a single value representing the heterozygosity between the two populations. For the four-population models, this entry includes six values. The first value is the heterozygosity between the first and the second population, the second value is between the first and the third population, the third value is between the first and fourth population, the fourth value is between the second and third populations, the fifth value is between the second and fourth population and the sixth value is between the third and fourth populations.

`SD_HetBet` numeric, standard deviation of the mean heterozygosity between all pairs of populations. For the two populations model, this is a single value representing the standard deviation of heterozygosity between the two populations. When running one of the four-population models, this entry includes six values. The order of those entries is the same as for `Mean_HetBet`.

`Mean_FST` numeric, mean pairwise FST between populations. For the two populations model, this is a single value representing the mean FST between the two populations. For the four-population models, this entry includes six values. The first value is the mean FST between the first and second populations, the second is between the first and third population, the third is between the second and third populations, the fourth is between the first and fourth populations, the fifth value is between the second and fourth populations and the sixth is between the third and fourth populations.

`SD_FST` numeric, standard deviation of the mean pairwise FST between populations. For the two populations model, this is a single value representing the standard deviation of the FST between the two populations. When running one of the four-population models, this entry includes six values. The order of those entries is the same as for `Mean_FST`.

FSTQ1 numeric, it is the 5% quantile of the mean pairwise FST distribution. For the two populations model, this is a single value representing the 5% quantile of the FST between the two populations. When running one of the four-population models, this entry includes six values. The order of those entries is the same as for Mean_FST.

FSTQ2 numeric, it is the 95% quantile of the mean pairwise FST distribution. For the two populations model, this is a single value representing the 95% quantile of the FST between the two populations. For the four-population models, this entry includes six values. The order of those entries is the same as for Mean_FST.

Dstat numeric, value of D-statistic for various combinations of populations. This entry only exists if a four-population model was selected. It includes three different values. For the first value, P1 was the W ecotype in the first location P2 was the W ecotype in the second location and P3 was the C ecotype at the first location. For the second value P1 was again the W ecotype in the first location but P2 was the C ecotype in the second ecotype and P3 was the C ecotype at the first location. For the third value, P1 was also the W ecotype at the first location, P2 was the C ecotype at the first location and P3 was the W ecotype at the second location. For all combinations, P4 was assumed to be an outgroup fixed, at all sites, for the major allele.

SD_dstat numeric, standard deviation of D-statistic for various combinations of populations. This entry only exists if a four-population model was selected. Each entry is the standard deviation of the corresponding D-statistic in the Dstat entry.

Examples

```
# simulate Pool-seq data and compute summary statistics for a model with two populations
poolSim(model="2pops", nDip=400, nPops=2, nLoci=10, nSites=2000, mutrate=1.5e-8,
size=rep(list(rep(5, 20)), 2),mean=c(85, 65), variance=c(1400, 900), minimum=25,
maximum=165, min.minor=2, Nref=c(25000, 25000), ratio=c(0.1, 3), pool=c(5, 250),
seq=c(0.0001, 0.001), split=c(0, 3), CW=c(1e-13, 1e-3), WC=c(1e-13, 1e-3), bT=c(0, 0.5))

# simulate Pool-seq data and compute summary statistics for a model with four populations
poolSim(model="Single", nDip=400, nPops=4, nLoci=10, nSites=2000, mutrate=2e-8,
size=rep(list(rep(5, 20)), 4), mean=c(85, 65, 65, 70), variance=c(1400, 900, 850, 1000),
minimum=25, maximum=165, min.minor=2, Nref=c(25000, 25000), ratio=c(0.1, 3), pool=c(5, 250),
seq=c(0.0001, 0.001), split=c(0, 3), CW=c(1e-13, 1e-3), WC=c(1e-13, 1e-3), CC=c(1e-13, 1e-3),
WW=c(1e-13, 1e-3), ANC=c(1e-13, 1e-3), bT=c(0, 0.2), bCW=c(0, 0.5), bWC=c(0, 0.5))
```

---

poolStats *Compute summary statistics from Pooled DNA sequencing*

Description

This function combines all the necessary steps to simulate pooled sequencing data and compute summary statistics from that data.

Usage

```
poolStats(
  parameters,
  model,
  nDip,
  size,
  nLoci,
  nSites,
  mutrate,
  mean,
  variance,
  minimum,
  maximum,
  min.minor = NA,
  force = FALSE
)
```

Arguments

parameters	a vector of parameters used to create the command line for the scrm package. Each entry of the vector is a different parameter. Note that each vector entry should be named with the name of the corresponding parameter. The output of the `CreateParameters` function is the intended input.
model	a character, either 2pops", "Single" or "Parallel" indicating which model should be simulated.
nDip	an integer representing the total number of diploid individuals to simulate. Note that scrm actually simulates haplotypes, so the number of simulated haplotypes is double of this. Also note that this is the total number of diploid individuals and this function will distribute the individuals equally by the simulated populations.
size	a list with one entry per population. Each entry should be a vector containing the size (in number of diploid individuals) of each pool. Thus, if a population was sequenced using a single pool, the vector should contain only one entry. If a population was sequenced using two pools, each with 10 individuals, this vector should contain two entries and both will be 10.
nLoci	an integer that represents how many independent loci should be simulated.
nSites	is an integer that specifies how many base pairs should scrm simulate, i.e. how many sites per locus to simulate.
mutrate	an integer representing the mutation rate assumed for the simulations.
mean	an integer or a vector defining the mean value of the negative binomial distribution from which different number of reads are drawn. It represents the mean coverage across all sites. If a vector is supplied, the function assumes that each entry of the vector is the mean for a different population.
variance	an integer or a vector defining the variance of the negative binomial distribution from which different number of reads are drawn. It represents the variance of the total coverage across all sites. If a vector is supplied, the function assumes that each entry of the vector is the variance for a different population.

`minimum` an integer representing the minimum coverage allowed. Sites where any population has a depth of coverage below this threshold are removed from the data.

`maximum` an integer representing the maximum coverage allowed. Sites where any population has a depth of coverage above this threshold are removed from the data.

`min.minor` is an integer representing the minimum allowed number of minor-allele reads. Sites that, across all populations, have less minor-allele reads than this threshold will be removed from the data.

`force` is a logical value indicating whether the required number of loci should be enforced. The default is FALSE but, if set to TRUE, then additional loci will be simulated. These additional loci are simulated to try to have sufficient loci to keep the required number of loci after filtering.

Details

The sampled parameter values are incorporated into a command line for the scrm package. Then, genetic data is simulated according to a model of ecotype formation and the sampled parameters. Finally, various summary statistics are calculated from the simulated data.

Value

a list with several named entries. The number of entries depends of the chosen model.

`nPoly` numeric, mean number of polymorphic sites across all simulated locus.

`nFilter` numeric, mean number of polymorphic sites retained after filtering across all simulated locus.

`nLoci` numeric, total number of loci retained after filtering. Summary statistics are calculated for these loci.

`Sf` numeric, fraction of sites fixed between populations. For the model with two populations, this is a single value. For the four-population models, this includes three values: the first is the fraction of fixed sites between the two populations in the first location, the second value is between the populations in the second location and the third value is the overall fraction of fixed sites, obtained by comparing each population against the other three.

`Sx` numeric, fraction of exclusive sites per population. When running the model with two populations, this entry has two values - one per population. For the four-population models, there is also one value per population, followed by a fifth value representing the fraction of sites that are segregating in only one of the populations.

`SS` numeric values representing the fraction of sites shared between populations. For the model with two populations, this is a single value. When running one of the four-population models, this entry has three values. The first is the fraction of shared sites between the two populations in the first location, the second value is between the populations in the second location and the third value is the fraction of shared sites across all four populations.

`Mean_Het` numeric, expected heterozygosity within each population. This entry has two values when using a two populations model and four when running one of the four-populations model.

`SD_Het` numeric, standard deviation of the expected heterozygosity for each population. This entry has two values when using a two populations model and four when running one of the four-populations model.

`Mean_HetBet` numeric, mean heterozygosity between all pairs of populations. For the two populations model, this is a single value representing the heterozygosity between the two populations. For the four-population models, this entry includes six values. The first value is the heterozygosity between the first and the second population, the second value is between the first and the third population, the third value is between the first and fourth population, the fourth value is between the second and third populations, the fifth value is between the second and fourth population and the sixth value is between the third and fourth populations.

`SD_HetBet` numeric, standard deviation of the mean heterozygosity between all pairs of populations. For the two populations model, this is a single value representing the standard deviation of heterozygosity between the two populations. When running one of the four-population models, this entry includes six values. The order of those entries is the same as for `Mean_HetBet`.

`Mean_FST` numeric, mean pairwise FST between populations. For the two populations model, this is a single value representing the mean FST between the two populations. For the four-population models, this entry includes six values. The first value is the mean FST between the first and second populations, the second is between the first and third population, the third is between the second and third populations, the fourth is between the first and fourth populations, the fifth value is between the second and fourth populations and the sixth is between the third and fourth populations.

`SD_FST` numeric, standard deviation of the mean pairwise FST between populations. For the two populations model, this is a single value representing the standard deviation of the FST between the two populations. When running one of the four-population models, this entry includes six values. The order of those entries is the same as for `Mean_FST`.

`FSTQ1` numeric, it is the 5% quantile of the mean pairwise FST distribution. For the two populations model, this is a single value representing the 5% quantile of the FST between the two populations. When running one of the four-population models, this entry includes six values. The order of those entries is the same as for `Mean_FST`.

`FSTQ2` numeric, it is the 95% quantile of the mean pairwise FST distribution. For the two populations model, this is a single value representing the 95% quantile of the FST between the two populations. For the four-population models, this entry includes six values. The order of those entries is the same as for `Mean_FST`.

`Dstat` numeric, value of D-statistic for various combinations of populations. This entry only exists if a four-population model was selected. It includes three different values. For the first value, P1 was the W ecotype in the first location P2 was the W ecotype in the second location and P3 was the C ecotype at the first location. For the second value P1 was again the W ecotype in the first location but P2 was the C ecotype in the second ecotype and P3 was the C ecotype at the first location. For the third value, P1 was also the W ecotype at the first location, P2 was the C ecotype at the first location and P3 was the W ecotype at the second

location. For all combinations, P4 was assumed to be an outgroup fixed, at all sites, for the major allele.

`SD_dstat` numeric, standard deviation of D-statistic for various combinations of populations. This entry only exists if a four-population model was selected. Each entry is the standard deviation of the corresponding D-statistic in the `Dstat` entry.

Examples

```
# create a vector of parameters for a model with two populations
parameters <- createParams(Nref = c(25000, 25000), ratio = c(0.1, 3), pool = c(5, 250),
seq = c(0.0001, 0.001), split = c(0, 3), CW = c(1e-13, 1e-3), WC = c(1e-13, 1e-3),
bT = c(0, 0.2), model = "2pops")

# simulate a two populations model:
# note that we are using two pools for each population, each with 50 individuals
poolStats(parameters = parameters, model = "2pops", nDip = 200, size = rep(list(rep(50, 2)), 2),
nLoci = 100, nSites = 2000, mutrate = 2e-8, mean = c(100, 80), variance = c(200, 180), minimum = 10,
maximum = 150, min.minor = 1)
```

---

poststat *Calculate point estimates from the posterior distribution*

Description

Given a set of samples from the posterior distribution, computes the mean, median and mode of the posterior.

Usage

```
poststat(posterior, limits, method, wtreg = NULL)
```

Arguments

`posterior` is a matrix or a vector with samples from the posterior distribution obtained from ABC parameter estimation. If this input is a matrix, then each row should correspond to an accepted simulation (size S) and each column to a different parameter.

`limits` is a vector if there is only one parameter or a matrix if there are multiple parameters. In this latter instance, each row should correspond to a different parameter. In either instance, and considering matrix rows as vectors, then the first entry of the vector should be the minimum value of the prior and the second entry should be the maximum value (for any given parameter).

`method` either "rejection" or "regression" indicating whether a rejection sampling algorithm or a local linear regression algorithm were used during ABC parameter estimation.

`wtreg` is a required numeric vector if the method is "regression". It should contain the weights for each accepted simulation (size S).

Details

If method is "regression", the regression weights must also be made available. These will be used to compute the weighted mean, weighted median and weighted mode of the posterior.

Value

a matrix with the mode, median and mean of the posterior distribution for each parameter. Each point estimate is a different row and each parameter a different column.

Examples

```
# load the matrix with parameter values
data(params)
# load the matrix with simulated parameter values
data(sumstats)
# load the matrix with the prior limits
data(limits)

# select a random simulation to act as target just to test the function
target <- sumstats[10 ,]
# we should remove the random simulation from the sumstats and params matrices
sumstats <- sumstats[-10, ]; params <- params[-10, ]

# parameter estimation for a single target
myabc <- singleABC(target = target, params = params, sumstats = sumstats,
limits = limits, tol = 0.01, method = "regression")

# compute point estimates from the posterior distribution
poststat(posterior = myabc$adjusted, limits = limits, method = "regression", wtreg = myabc$wt)
```

---

prepareData *Organize information by contig - for multiple data files*

Description

Organize the information of multiple _rc files into different entries for each contig.

Usage

```
prepareData(data, nPops, filter = FALSE, threshold = NA)
```

Arguments

data	is a list with four different entries. The entries should be named as "rMajor", "rMinor", "coverage" and "info". The rMajor entry should be a matrix containing the number of observed major-allele reads. The rMinor entry should be a matrix containing the number of observed minor-allele reads. The coverage entry should be a matrix containing the total depth of coverage. The info entry

should be a matrix or a data frame containing the remaining relevant information, such as the contig name and the position of each SNP. Each row of these matrices should be a different site and each column should be a different population.

`nPops` is an integer indicating the total number of different populations in the dataset.

`filter` is a logical switch, either TRUE or FALSE. If TRUE, then the data is filtered by the frequency of the minor allele and if FALSE, that filter is not applied.

`threshold` is the minimum allowed frequency for the minor allele. Sites where the allelic frequency is below this threshold are removed from the data.

Details

This function removes all monomorphic sites from the dataset. Monomorphic sites are those where the frequency for all populations is 1 or 0. Then, the name of each contig is used to organize the information in a per contig basis. Thus, each output will be organized by contig. For example, the list with the number of minor-allele reads will contain several entries and each of those entries is a different contig.

If the filter input is set to TRUE, this function also filters the data by the frequency of the minor-allele. If a threshold is supplied, the computed frequency is compared to that threshold and sites where the frequency is below the threshold are removed from the dataset. If no threshold is supplied, the threshold is assumed to be `1/total coverage`, meaning that a site should have, at least, one minor-allele read.

Value

a list with six named entries:

`freqs` a list with the allele frequencies, computed by dividing the number of minor-allele reads by the total coverage. Each entry of this list corresponds to a different contig. Each entry is a matrix where each row is a different site and each column is a different population.

`positions` a list with the positions of each SNP. Each entry of this list is a vector corresponding to a different contig.

`range` a list with the minimum and maximum SNP position of each contig. Each entry of this list is a vector corresponding to a different contig.

`rMajor` a list with the number of major-allele reads. Each entry of this list corresponds to a different contig. Each entry is a matrix where each row is a different site and each column is a different population.

`rMinor` a list with the number of minor-allele reads. Each entry of this list corresponds to a different contig. Each entry is a matrix where each row is a different site and each column is a different population.

`coverage` a list with the total coverage. Each entry of this list corresponds to a different contig. Each entry is a matrix where each row is a different site and each column is a different population.

Examples

```
# load the data from two rc files
data(rc1, rc2)
# combine both files into a single list
mydata <- list(rc1, rc2)

# clean and organize the data for both files
mydata <- lapply(mydata, function(i) cleanData(file = i, pops = 7:10))

# organize the information by contigs
prepareData(data = mydata, nPops = 4)
```

---

prepareFile *Organize information by contigs - for a single data file*

Description

Organize the information of a single _rc file into different entries for each contig.

Usage

```
prepareFile(data, nPops, filter = FALSE, threshold = NA)
```

Arguments

data	is a list with four different entries. The entries should be named as "rMajor", "rMinor", "coverage" and "info". The rMajor entry should be a matrix containing the number of observed major-allele reads. The rMinor entry should be a matrix containing the number of observed minor-allele reads. The coverage entry should be a matrix containing the total depth of coverage. The info entry should be a matrix or a data frame containing the remaining relevant information, such as the contig name and the position of each SNP. Each row of these matrices should be a different site and each column should be a different population.
nPops	is an integer indicating the total number of different populations in the dataset.
filter	is a logical switch, either TRUE or FALSE. If TRUE, then the data is filtered by the frequency of the minor allele and if FALSE, that filter is not applied.
threshold	is the minimum allowed frequency for the minor allele. Sites where the allelic frequency is below this threshold are removed from the data.

Details

This function removes all monomorphic sites from the dataset. Monomorphic sites are those where the frequency for all populations is 1 or 0. Then, the name of each contig is used to organize the information in a per contig basis. Thus, each output will be organized by contig. For example, the

list with the number of minor-allele reads will contain several entries and each of those entries is a different contig.

If the filter input is set to TRUE, this function also filters the data by the frequency of the minor-allele. If a threshold is supplied, the computed frequency is compared to that threshold and sites where the frequency is below the threshold are removed from the dataset. If no threshold is supplied, the threshold is assumed to be `1/total coverage`, meaning that a site should have, at least, one minor-allele read.

Value

a list with six named entries:

`freqs`	a list with the allele frequencies, computed by dividing the number of minor-allele reads by the total coverage. Each entry of this list corresponds to a different contig. Each entry is a matrix where each row is a different site and each column is a different population.
`positions`	a list with the positions of each SNP. Each entry of this list is a vector corresponding to a different contig.
`range`	a list with the minimum and maximum SNP position of each contig. Each entry of this list is a vector corresponding to a different contig.
`rMajor`	a list with the number of major-allele reads. Each entry of this list corresponds to a different contig. Each entry is a matrix where each row is a different site and each column is a different population.
`rMinor`	a list with the number of minor-allele reads. Each entry of this list corresponds to a different contig. Each entry is a matrix where each row is a different site and each column is a different population.
`coverage`	a list with the total coverage. Each entry of this list corresponds to a different contig. Each entry is a matrix where each row is a different site and each column is a different population.

Examples

```
# load the data from one rc file
data(rc1)

# clean and organize the data in this single file
mydata <- cleanData(file = rc1, pops = 7:10)

# organize the information by contigs
prepareFile(data = mydata, nPops = 4)
```

priorsMatrix *Construct matrix of prior limits*

Description

Takes as input the minimum and maximum values of the prior distribution for all relevant parameters and constructs a matrix of prior limits.

Usage

```
priorsMatrix(model, inputParams)
```

Arguments

model	a character, either 2pops", "Single" or "Parallel" indicating which model was simulated.
inputParams	A vector containing the minimum and maximum values of the prior distribution for each parameter in the model. The input of the CreateParameters function can be converted into a vector and used here.

Details

The output matrix contains all parameters of a given model and, for each parameter, it contains the minimum and maximum value of the prior.

Value

a matrix where each row is a different parameter. Note also that each row is named after the corresponding parameter. For each row, the first column contains the minimum value of that parameter and the second column contains the maximum value.

Examples

```
# create a vector of input parameters for a model with two populations
inputs <- c(Nref = c(25000, 25000), ratio = c(0.1, 3), pool = c(5, 250), seq = c(0.0001, 0.001),
split = c(0, 3), CW = c(1e-13, 1e-3), WC = c(1e-13, 1e-3), bT = c(0, 0.2))

# construct a matrix with the limits of the prior distribution
priorsMatrix(model = "2pops", inputParams = inputs)
```

rc1 *Data frame with an example of observed data*

Description

Data frame with data in the _rc format for 25 populations Each row of the data frame is a different site. The first 9 columns contain general information about the site, while the remaining contain the number of reads observed at that site for each of the 25 populations.

Usage

```
rc1
```

Format

a data frame with 5000 rows and 59 columns. Each of the columns corresponds to :

col1 reference chromosome (contig).

col2 reference position.

col3 reference character.

col4 number of alleles found in all populations.

col5 allele characters in all populations (sorted by counts in all populations).

col6 sum of deletions in all populations (should be zero, if not the position may not be reliable).

col7 SNP type: [pop, rc, rc|pop]; pop: a SNP within or between the populations; rc: a SNP between the reference sequence character and the consensus of at least one populaton; rc|pop: both.

col8 most frequent allele in all populations [12345..].

col9 second most frequent allele in all populations [12345..].

col10 - col34 frequencies of the most frequent allele (major) in the form "allele-count/coverage.

col35 - col59 frequencies of the second most frequent allele (minor) in the form "allele-count/coverage".

Source

Hernán E. Morales et al., Genomic architecture of parallel ecological divergence: Beyond a single environmental contrast. Sci. Adv.5, eaav9963(2019). DOI:10.1126/sciadv.aav9963

rc2 *Data frame with an example of observed data*

Description

Data frame with data in the _rc format for 25 populations Each row of the data frame is a different site. The first 9 columns contain general information about the site, while the remaining contain the number of reads observed at that site for each of the 25 populations.

Usage

```
rc2
```

Format

a data frame with 5000 rows and 59 columns. Each of the columns corresponds to :

col1 reference chromosome (contig).

col2 reference position.

col3 reference character.

col4 number of alleles found in all populations.

col5 allele characters in all populations (sorted by counts in all populations).

col6 sum of deletions in all populations (should be zero, if not the position may not be reliable).

col7 SNP type: [pop, rc, rc|pop]; pop: a SNP within or between the populations; rc: a SNP between the reference sequence character and the consensus of at least one populaton; rc|pop: both.

col8 most frequent allele in all populations [12345..].

col9 second most frequent allele in all populations [12345..].

col10 - col34 frequencies of the most frequent allele (major) in the form "allele-count/coverage.

col35 - col59 frequencies of the second most frequent allele (minor) in the form "allele-count/coverage".

Source

Hernán E. Morales et al., Genomic architecture of parallel ecological divergence: Beyond a single environmental contrast. Sci. Adv.5, eaav9963(2019). DOI:10.1126/sciadv.aav9963

regABC *Parameter estimation with Approximate Bayesian Computation using local linear regression*

Description

This function performs multivariate parameter estimation based on summary statistics using an Approximate Bayesian Computation (ABC) algorithm. The algorithm used here is the local linear regression algorithm.

Usage

```
regABC(rej, parameter, tol = 1, simple = FALSE)
```

Arguments

`rej`	is a list with the results of the rejection sampling algorithm. The output of the `rejABC()` function is the ideal input here.
`parameter`	is a parameter vector (long vector of numbers from the simulations). Each vector entry should correspond to a different simulation. This is the dependent variable for the regression.
`tol`	is the tolerance rate, indicating the required proportion of points accepted nearest the target values. Note that the default value here is 1 because all points accepted in the rejection step should be used for the regression.
`simple`	logical, if TRUE a simplified output with only the essential information will be produced. If FALSE (default) the output will contain more information.

Details

Note that to use this function, the usual steps of ABC parameter estimation have to be performed. Briefly, data should have been simulated based on random draws from the prior distributions of the parameters of interest and a set of summary statistics should have been calculated from that data. The same set of summary statistics should have been calculated from the observed data to be used as the target for parameter inference. A previous rejection sampling step should also have been performed, where parameter values were accepted if the Euclidean distance between the set of summary statistics computed from the simulated data and the set of summary statistics computed from the observed data was sufficiently small. Then, the output of the rejection step is used as the input for this function and a local linear regression method is used to correct for the imperfect match between the summary statistics computed from the simulated data and the summary statistics computed from the observed data.

The parameter values accepted in the rejection step are weighted by a smooth function (kernel) of the distance between the simulated and observed summary statistics and corrected according to a linear transformation. This function calls the function `stats::lm()` to accomplish this.

Value

a list with the results from the regression correction

adjusted	regression adjusted parameter values.
unadjusted	parameter estimates obtained with the rejection sampling.
wt	regression weights.
ss	set of accepted summary statistics from the simulations.
predmean	estimates of the posterior mean for each parameter.
fv	fitted value from the regression.

Examples

```
# load the matrix with parameter values
data(params)
# load the matrix with simulated parameter values
data(sumstats)

# select a random simulation to act as target just to test the function
target <- sumstats[10 ,]

# parameter estimation using rejection sampling
rej <- rejABC(target = target, params = params, sumstats = sumstats[-10, ],
tol = 0.01, regression = TRUE)

# parameter estimation using local linear regression
# note that you should select a parameter from the unadjusted matrix
regABC(rej = rej, parameter = rej$unadjusted[, 1])
```

---

rejABC *Parameter estimation with Approximate Bayesian Computation using rejection sampling*

Description

This function performs multivariate parameter estimation based on summary statistics using an Approximate Bayesian Computation (ABC) algorithm. The algorithm used here is the rejection sampling algorithm. The output of this function can be tailored towards a posterior local linear regression method correction.

Usage

```
rejABC(target, params, sumstats, tol, regression = FALSE)
```

Arguments

target	a vector with the target summary statistics. These are usually the set of observed summary statistics.
params	is a vector or matrix of simulated parameter values i.e. numbers from the simulations. Each row or vector entry should be a different simulation and each column of a matrix should be a different parameter.
sumstats	is a vector or matrix of simulated summary statistics. Each row or vector entry should be a different simulation and each column of a matrix should be a different statistic.
tol	is the tolerance rate, indicating the required proportion of points accepted nearest the target values.
regression	logical, indicating whether the user intends to perform a local linear regression correction after the rejection step. If set to FALSE (default) the output of this function will contain just the results of the rejection step. If set to TRUE, the output will contain more details required for the regression step.

Details

The rejection sampling algorithm generates random samples from the posterior distributions of the parameters of interest. Note that to use this function, the usual steps of ABC parameter estimation have to be performed. Briefly, data should have been simulated based on random draws from the prior distributions of the parameters of interest and a set of summary statistics should have been calculated from that data. The same set of summary statistics should have been calculated from the observed data to be used as the `target` input in this function. Parameter values are accepted if the Euclidean distance between the set of summary statistics computed from the simulated data and the set of summary statistics computed from the observed data is sufficiently small. The percentage of accepted simulations is determined by `tol`.

Value

a list with the results of the rejection sampling algorithm. The elements of the list depend of the logical value of `regression`.

s.target	a scaled vector of the observed summary statistics. This element only exists if regression is TRUE.
unadjusted	parameter estimates obtained with the rejection sampling.
ss	set of accepted summary statistics from the simulations.
s.sumstat	set of scaled accepted summary statistics from the simulations. This element only exists if regression is TRUE.
dst	euclidean distances in the region of interest.

Examples

```
# load the matrix with parameter values
data(params)
# load the matrix with simulated parameter values
data(sumstats)
```

```
# select a random simulation to act as target just to test the function
target <- sumstats[10 ,]

# Parameter estimation using rejection sampling
rejABC(target = target, params = params, sumstats = sumstats[-10, ], tol = 0.01)
```

`remove_quantileReads` *Remove sites using quantiles of the depth of coverage*

Description

Removes sites that have too many or too few reads from the dataset.

Usage

```
remove_quantileReads(nPops, data)
```

Arguments

`nPops`	is an integer representing the total number of populations in the dataset.
`data`	is a dataset containing information about real populations. This dataset should have lists with the allelic frequencies, the position of the SNPs, the range of the contig, the number of major allele reads, the number of minor allele reads and the depth of coverage.

Details

The 25% and the 75% quantiles of the coverage distribution is computed for each population in the dataset. Then, the lowest 25% quantile across all populations is considered the minimum depth of coverage allowed. Similarly, the highest 75% quantile across all populations is considered the maximum depth of coverage allowed. The coverage of each population at each site is compared with those threshold values and any site, where the coverage of at least one population is below or above that threshold, is completely removed from the dataset.

Value

a list with the following elements:

`freqs`	a list with the allele frequencies, computed by dividing the number of minor-allele reads by the total coverage. Each entry of this list corresponds to a different contig. Each entry is a matrix where each row is a different site and each column is a different population.
`positions`	a list with the positions of each SNP. Each entry of this list is a vector corresponding to a different contig.
`range`	a list with the minimum and maximum SNP position of each contig. Each entry of this list is a vector corresponding to a different contig.

rMajor	a list with the number of major-allele reads. Each entry of this list corresponds to a different contig. Each entry is a matrix where each row is a different site and each column is a different population.
rMinor	a list with the number of minor-allele reads. Each entry of this list corresponds to a different contig. Each entry is a matrix where each row is a different site and each column is a different population.
coverage	a list with the total coverage. Each entry of this list corresponds to a different contig. Each entry is a matrix where each row is a different site and each column is a different population.

This output is identical to the data input, the only difference being the removal of sites with too many or too few reads.

Examples

```
# load the data from one rc file
data(rc1)

# clean and organize the data in this single file
mydata <- cleanData(file = rc1, pops = 7:10)

# organize the information by contigs
mydata <- prepareFile(data = mydata, nPops = 4)

# remove sites according to the coverage quantile
remove_quantileReads(nPops = 4, data = mydata)
```

---

remove_realReads *Remove sites, according to their coverage, from real data*

Description

Removes sites that have too many or too few reads from the dataset.

Usage

```
remove_realReads(nPops, data, minimum, maximum)
```

Arguments

nPops	is an integer representing the total number of populations in the dataset.
data	is a dataset containing information about real populations. This dataset should have lists with the allelic frequencies, the position of the SNPs, the range of the contig, the number of major allele reads, the number of minor allele reads and the depth of coverage.
minimum	the minimum depth of coverage allowed i.e. sites where the depth of coverage of any population is below this threshold are removed.

`maximum` he maximum depth of coverage allowed i.e. sites where the depth of coverage of any population is above this threshold are removed.

Details

The `minimum` and `maximum` inputs define, respectively, the minimum and maximum allowed coverage for the dataset. The coverage of each population at each site is compared with those threshold values and any site, where the coverage of at least one population is below or above the user defined threshold, is completely removed from the dataset.

Value

a list with the following elements:

`freqs` a list with the allele frequencies, computed by dividing the number of minor-allele reads by the total coverage. Each entry of this list corresponds to a different contig. Each entry is a matrix where each row is a different site and each column is a different population.

`positions` a list with the positions of each SNP. Each entry of this list is a vector corresponding to a different contig.

`range` a list with the minimum and maximum SNP position of each contig. Each entry of this list is a vector corresponding to a different contig.

`rMajor` a list with the number of major-allele reads. Each entry of this list corresponds to a different contig. Each entry is a matrix where each row is a different site and each column is a different population.

`rMinor` a list with the number of minor-allele reads. Each entry of this list corresponds to a different contig. Each entry is a matrix where each row is a different site and each column is a different population.

`coverage` a list with the total coverage. Each entry of this list corresponds to a different contig. Each entry is a matrix where each row is a different site and each column is a different population.

This output is identical to the `data` input, the only difference being the removal of sites with too many or too few reads.

Examples

```
# load the data from one rc file
data(rc1)

# clean and organize the data in this single file
mydata <- cleanData(file = rc1, pops = 7:10)

# organize the information by contigs
mydata <- prepareFile(data = mydata, nPops = 4)

# remove sites with less than 10 reads or more than 180
remove_realReads(nPops = 4, data = mydata, minimum = 10, maximum = 180)
```

runSCRM *Run scrm and obtain genotypes*

Description

This function will run the scrm package, according to the command line supplied as input. It will also combine haplotypes into genotypes and re-organize the output if the simulations were performed under a single origin scenario. This is to ensure that the output of the four-population models will always follow the same order: the two divergent ecotypes in the first location, followed by the two divergent ecotypes in the second location.

Usage

```
runSCRM(commands, nDip, nPops, model)
```

Arguments

commands	A character string containing the commands for the scrm package. This string can be created using the cmd2pops, the cmdSingle or the cmdParallel functions.
nDip	An integer representing the total number of diploid individuals to simulate. Note that scrm actually simulates haplotypes, so the number of simulated haplotypes is double of this.
nPops	An integer that informs of how many populations exist on the model you are trying to run.
model	Either "2pops", "Single" or "Parallel" indicating which model should be simulated.

Value

a list with the simulated genotypes. Each entry is a different locus and, for each locus, different rows represent different individuals and each column is a different site.

Examples

```
# create a vector with parameter values for a two populations model
params <- createParams(Nref = c(25000, 25000), ratio = c(0.1, 3), pool = c(5, 250),
seq = c(0.0001, 0.001), split = c(0, 3), CW = c(1e-13, 1e-3), WC = c(1e-13, 1e-3),
bT = c(0, 0.2), model = "2pops")

# create the command line for the scrm package
cmds <- cmd2pops(parameters = params, nSites = 2000, nLoci = 10, nDip = 100, mutrate = 2e-8)

# run SCRM and obtain the genotypes
runSCRM(commands = cmds, nDip = 100, nPops = 2, model = "2pops")
```

`scaled.migration` *Compute scaled migration rates*

Description

Computes and adds scaled migration rates to a matrix of simulated parameter values.

Usage

```
scaled.migration(parameters, model, Nref = NA)
```

Arguments

`parameters`	is a matrix of simulated parameter values i.e. numbers from the simulations. Each row or vector entry should be a different simulation and each column of a matrix should be a different parameter.
`model`	a character, either 2pops", "Single" or "Parallel" indicating which model was simulated.
`Nref`	a numeric value indicating the effective population size of the reference population.

Details

Migration rates are scaled according to the size of the population receiving the migrants and added to a matrix with the simulated parameter values. This is performed for the three available models and according to the specific model conformation.

Value

a matrix of simulated parameter values with added columns containing the scaled migration rates.

Examples

```
# compute scaled migration for a two-population model
scaled.migration(parameters = myparams, model = "2pops", Nref = 10000)
```

scaledPrior *Compute scaled migration rate limits*

Description

Computes and adds scaled migration rates to a matrix with the limits of the prior distributions.

Usage

```
scaledPrior(limits, model, Nref = NA)
```

Arguments

limits	is a matrix with two columns and as many rows as there are parameters. Each row should contain the minimum value of the prior for a given parameter in the first column and the maximum value in the second column.
model	a character, either 2pops", "Single" or "Parallel" indicating which model was simulated.
Nref	a numeric value indicating the effective population size of the reference population.

Details

Migration rates are scaled according to the size of the population receiving the migrants and added to a matrix with the prior limits. The minimum and maximum possible size of the population and of the migration rate are used to compute the minimum and maximum possible values of the scaled migration rates. This is performed for the three available models and according to the specific model conformation.

Value

a matrix where each row is a different parameter. This matrix is similar to the input argument limits but with added rows containing the scaled migration rates.

Examples

```
# create a vector of input parameters for a model with two populations
inputs <- c(Nref = c(25000, 25000), ratio = c(0.1, 3), pool = c(5, 250), seq = c(0.0001, 0.001),
split = c(0, 3), CW = c(1e-13, 1e-3), WC = c(1e-13, 1e-3), bT = c(0, 0.2))

# construct a matrix with the limits of the prior distribution
limits <- priorsMatrix(model = "2pops", inputParams = inputs)

# compute and add the prior limits of the scaled migration
scaledPrior(limits = limits, model = "2pops")
```

`simulationABC` *Perform an Approximate Bayesian Computation simulation study*

Description

Perform a leave-one-out cross validation for ABC via subsequent calls to the `singleABC()` function.

Usage

```
simulationABC(
  params,
  sumstats,
  limits,
  nval,
  tol,
  method,
  parallel = FALSE,
  ncores = NA
)
```

Arguments

`params`	is a vector or matrix of simulated parameter values i.e. numbers from the simulations. Each row or vector entry should be a different simulation and each column of a matrix should be a different parameter. This is the dependent variable for the regression, if a regression step is performed.
`sumstats`	is a vector or matrix of simulated summary statistics. Each row or vector entry should be a different simulation and each column of a matrix should be a different statistic. These act as the independent variables if a regression step is performed.
`limits`	is a matrix with two columns and as many rows as there are parameters. Each row should contain the minimum value of the prior for a given parameter in the first column and the maximum value in the second column.
`nval`	size of the cross-validation sample i.e. how many different evaluations should be performed. Each evaluation corresponds to a different target for the parameter estimation.
`tol`	is the tolerance rate, indicating the required proportion of points accepted nearest the target values.
`method`	either "rejection" or "regression" indicating whether a regression step should be performed during ABC parameter estimation.
`parallel`	logical, indicating whether this function should be run using parallel execution. The default setting is FALSE, meaning that this function will utilize a single core.
`ncores`	a non-negative integer that is required when `parallel` is TRUE. It specifies the number of cores to use for parallel execution.

Details

This function allows users to evaluate the impact of different tolerance rate on the quality of the estimation with ABC and whether a local linear regression algorithm improves the estimates. In subsequent steps, different point estimates of the posterior estimates can be compared with the true values, allowing the users to select the point estimate that leads to lower errors. Thus, performing a leave-one-out cross validation aids in selecting which point estimate is best - the mean, median or mode.

Value

a list with the following elements:

`true`	The parameter values of the simulations that served as validation.
`rej`	a list with the estimated parameter values under the rejection algorithm and using three different point estimates: mode, median and mean. The final entry of the list is the prediction error for each parameter, considering each of those point estimates as the estimated value.
`reg`	if method is "regression" then this is a list with the estimated parameter values under the regression algorithm and using three different point estimates: mode, median and mean. The final entry of the list is the prediction error for each parameter, considering each of those point estimates as the estimated value.

Examples

```
# load the matrix with parameter values
data(params)
# load the matrix with simulated parameter values
data(sumstats)
# load the matrix with the prior limits
data(limits)

# perform a leave-one-out cross validation for ABC
simulationABC(params = params, sumstats = sumstats, limits, nval = 10,
tol = 0.01, method = "regression")
```

---

sim_modelSel *Leave-one-out cross validation of model selection*

Description

This function performs a simulation study to assess the quality of model selection with ABC. This is done by performing a leave-one-out cross validation via subsequent calls to the function `modelSelect()`.

Usage

```
sim_modelSel(index, sumstats, nval, tol, warning = FALSE)
```

Arguments

index	is a vector of model indices. This can be a a character vector of model names, repeated as many times as there are simulations for each model. This vector will be coerced to factor and it must have the same length as nrow(sumstats) to indicate which row of the sumstats matrix belongs to which model.
sumstats	is a vector or matrix containing the simulated summary statistics for all the models. Each row or vector entry should be a different simulation and each column of a matrix should be a different statistic. The order must be the same as the order of the models in the index vector.
nval	a numerical value defining the the size of the cross-validation sample for each model.
tol	is a numerical value, indicating the required proportion of points nearest the target values (tolerance).
warning	logical, if FALSE (default) warnings produced while running this function, mainly related with accepting simulations for just one of the models, will not be displayed.

Details

One simulation is randomly selected from each model to be a validation simulation, while all the other simulations are used as training simulations. This random simulation is used as the target of the modelSelect() function and posterior model probabilities are estimated.

Please note that the actual size of the cross-validation sample is nval*the number of models. This is because nval cross-validation estimation steps are performed for each model.

Value

a list with the following elements:

cvsamples	is a vector of length nval*the number of models indicating which rows of the sumstat input were used as validation values for each model.
true	a character vector of the true models.
estimated	a character vector of the estimated models.
model.probs	a matrix with the estimated model probabilities. Each row of the matrix represents a different cross-validation trial.
models	a character vector with the designation of the models.

Examples

```
# load the matrix with simulated parameter values
data(sumstats)

# select a random simulation to act as target just to test the function
target <- sumstats[10 ,]

# create a "fake" vector of model indices
# this assumes that half the simulations were from one model and the other half from other model
```

```
# this is not true but serves as an example of how to use this function
index <- c(rep("model1", nrow(sumstats)/2), rep("model2", nrow(sumstats)/2))

# perform a leave-one-out cross validation of model selection
sim_modelSel(index = index, sumstats = sumstats, nval = 10, tol = 0.1)
```

`singleABC` *Parameter estimation with Approximate Bayesian Computation for a single target*

Description

Perform multivariate parameter estimation based on summary statistics using an Approximate Bayesian Computation (ABC) algorithm. This function always uses a rejection sampling algorithm while a local linear regression algorithm might or might not be used.

Usage

```
singleABC(target, params, sumstats, limits, tol, method)
```

Arguments

`target`	a vector with the target summary statistics. These are usually computed from observed data or selected from a random simulation when performing cross-validation.
`params`	is a vector or matrix of simulated parameter values i.e. numbers from the simulations. Each row or vector entry should be a different simulation and each column of a matrix should be a different parameter. This is the dependent variable for the regression, if a regression step is performed.
`sumstats`	is a vector or matrix of simulated summary statistics. Each row or vector entry should be a different simulation and each column of a matrix should be a different statistic. These act as the independent variables if a regression step is performed.
`limits`	is a matrix with two columns and as many rows as there are parameters. Each row should contain the minimum value of the prior for a given parameter in the first column and the maximum value in the second column.
`tol`	is the tolerance rate, indicating the required proportion of points accepted nearest the target values.
`method`	either "rejection" or "regression" indicating whether a regression step should be performed during ABC parameter estimation.

Details

To use this function, the usual steps of ABC parameter estimation have to be performed. Briefly, data should have been simulated based on random draws from the prior distributions of the parameters of interest and a set of summary statistics should have been calculated from that data. The same set of summary statistics should have been calculated from the observed data to be used as the `target` input in this function. Parameter values are accepted if the Euclidean distance between the set of summary statistics computed from the simulated data and the set of summary statistics computed from the observed data is sufficiently small. The percentage of accepted simulations is determined by `tol`. This function performs a simple rejection by calling the `rejABC()` function.

When `method` is "regression", a local linear regression method is used to correct for the imperfect match between the summary statistics computed from the simulated data and the summary statistics computed from the observed data. The output of the `rejABC()` function is used as the input of the `regABC()` function to apply this correction. The parameter values accepted in the rejection step are weighted by a smooth function (kernel) of the distance between the simulated and observed summary statistics and corrected according to a linear transformation.

Value

the returned object is a list containing the following components:

`unadjusted`	parameter estimates obtained with the rejection sampling.
`rej.prediction`	point estimates of the posterior obtained with the rejection sampling.
`adjusted`	regression adjusted parameter values.
`loc.prediction`	point estimates of the regression adjusted posterior.
`ss`	set of accepted summary statistics from the simulations.
`wt`	regression weights.

Examples

```
# load the matrix with parameter values
data(params)
# load the matrix with simulated parameter values
data(sumstats)
# load the matrix with the prior limits
data(limits)

# select a random simulation to act as target just to test the function
target <- sumstats[10 ,]
# we should remove the random simulation from the sumstats and params matrices
sumstats <- sumstats[-10, ]; params <- params[-10, ]

# parameter estimation for a single target
singleABC(target = target, params = params, sumstats = sumstats, limits = limits,
tol = 0.01, method = "regression")
```

summary_modelSelect *Posterior model probabilities*

Description

Extract the posterior model probabilities and obtain a summary of model selection performed with Approximate Bayesian Computation.

Usage

```
summary_modelSelect(object, print = TRUE)
```

Arguments

object	a list created by the modelSelect() function, containing results of model selection with Approximate Bayesian Computation.
print	logical, if TRUE (default), then this function prints the mean models probabilities.

Details

This function produces an easy-to-read output of the model selection step. It also computes the Bayes factors.

Value

a list with two main elements if model selection used the regression algorithm or a single element if only the rejection step was used:

rejection	results of model selection based on the rejection method. This element contains two entries, the first is an object of class numeric with the posterior model probabilities and the second are the Bayes factors between pairs of models.
mnlogistic	results of model selection based on the regression method. This element contains two entries, the first is an object of class numeric with the posterior model probabilities and the second are the Bayes factors between pairs of models.

Examples

```
# load the matrix with simulated parameter values
data(sumstats)

# select a random simulation to act as target just to test the function
target <- sumstats[10 ,]

# create a "fake" vector of model indices
# this assumes that half the simulations were from one model and the other half from other model
# this is not true but serves as an example of how to use this function
index <- c(rep("model1", nrow(sumstats)/2), rep("model2", nrow(sumstats)/2))
```

```
# perform model selection with ABC
mysel <- modelSelect(target = target, index = index, sumstats = sumstats,
tol = 0.01, method = "regression")

# compute posterior model probabilities
summary_modelSelect(object = mysel)
```

`sumstats` *Matrix of summary statistics computed from simulated data*

Description

This data set contains a set of 14 summary statistics computed from data simulated under an isolation with migration model of two populations.

Usage

```
sumstats
```

Format

a matrix with 10000 rows and 14 columns:

Sf fraction of sites fixed between populations.

Sx1 fraction of exclusive sites for the first population.

Sx2 fraction of exclusive sites for the second population.

SS fraction of sites shared between the two populations.

Mean_Het1 mean expected heterozygosity of the first population.

Mean_Het2 mean expected heterozygosity of the second population.

SD_Het1 standard deviation across loci of the mean expected heterozygosity of the first population.

SD_Het2 standard deviation across loci of the mean expected heterozygosity of the second population.

Mean_HetBet mean heterozygosity between the two populations.

SD_HetBet standard deviation across loci of the mean heterozygosity between the two populations.

Mean_FST mean pairwise FST between the two populations.

SD_FST standard deviation across loci of the mean pairwise FST between the two populations.

FSTQ1 5% quantile of the mean pairwise FST distribution.

FSTQ2 95% quantile of the mean pairwise FST distribution.

Source

simulations performed

Index

∗ **datasets**
limits, 21
myparams, 29
params, 30
rc1, 55
rc2, 56
sumstats, 72

ABC, 3
ABC(), *35*

cleanData, 6
cmd2pops, 7
cmdParallel, 8
cmdSingle, 10
createHeader, 11
createParams, 12

error_modelSel, 14
error_modelSel(), *32*

forceLocus, 15
forcePool, *17*

getmode, 17

importContigs, 18
index.rejABC, 20

limits, 21
locfit::locfit(), *18*

mergepost, 22
mergepost(), *37, 38*
mode_locfit, 26
mode_locfit(), *23*
modelSelect, 24
modelSelect(), *67, 68*, *71*
multinom, *25*
multipleABC, 27
multipleABC(), *22*, *34, 35*, *37*

myparams, 29

params, 30
plot_errorABC, 30
plot_msel, 32
plot_param, 33
plot_Posteriors, 34
plot_stats, 36
plot_weighted, 37
poolSim, 39
poolStats, 45
poststat, 49
prepareData, 50
prepareFile, 52
priorsMatrix, 54

rc1, 55
rc2, 56
regABC, 57
regABC(), *5*, *28*, *70*
rejABC, 58
rejABC(), *5*, *20*, *28*, *57*, *70*
remove_quantileReads, 60
remove_realReads, 61
runSCRM, 63

scaled.migration, 64
scaledPrior, 65
sim_modelSel, 67
sim_modelSel(), *14*
simulationABC, 66
singleABC, 69
singleABC(), *34*, *66*
stats::lm(), *57*
stats::predict(), *18*, *26*
summary_modelSelect, 71
sumstats, 72

weighted_stats(), *23*