Package 'strataG'

April 11, 2017

Title Summaries and Population Structure Analyses of Genetic Data

Description A toolkit for analyzing stratified population genetic data. Functions are provided for summarizing and checking loci (haploid, diploid, and polyploid), single stranded DNA sequences, calculating most population subdivision metrics, and running external programs such as structure and fastsimcoal. The package is further described in Archer et al (2016) <doi:10.1111/1755-0998.12559>.

Version 2.0.2

License GNU General Public License

URL https://github.com/EricArcher/strataG

BugReports https://github.com/EricArcher/strataG/issues

Depends R (>= 3.2.0), apex, ape, adegenet

Suggests knitr, rmarkdown, testthat

Imports data.table, copula, DT, ggplot2, graphics, grid, gridExtra, Hmisc, methods, pegas, phangorn, RColorBrewer, Rcpp, shiny, shinyFiles, stats, survival (>= 2.40.1), swfscMisc, utils

Collate strataG-package.R gtypes.class.R gtypes.accessors.R is.gtypes.R initialize.gtypes.R strataG-internal.R as.array.gtypes.R as.matrix.gtypes.R as.data.frame.gtypes.R alleleFreqFormat.R alleleFreqs.R alleleSplit.R allelicRichness.R arlequin.R as.multidna.R baseFreqs.R clumpp.R createConsensus.R dupGenotypes.R evanno.R expandHaplotypes.R fasta.R fastsimcoal.input.R fastsimcoal.R fixedDifferences.R fixedSites.R freq2GenData.R fusFs.R gelato.R genepop.R heterozygosity.R hweTest.R iupacCode.R jackHWE.R labelHaplotypes.R ldNe.R LDgenepop.R lowFreqSubs.R mRatio.R maf.R mafft.R maverickRun.R mega.R mostDistantSequences.R mostRepresentativeSequences.R neiDa.R nucleotideDivergence.R nucleotideDiversity.R numAlleles.R numericSNPmat.R numGenotyped.R numMissing.R permuteStrata.R phase.R popGenEqns.R popStructStat.R popStructTest.R privateAlleles.R propUniqueAlleles.R qaqc.R RcppExports.R readGenData.R removeSequences.R sequenceLikelihoods.R sharedLoci.R

show.gtypes.R simGammaHaps.R strataGUI.R strataSplit.R
stratify.R structure.R structurePlot.R summarizeLoci.R
summarizeSamples.R summarizeSeqs.R summary.gtypes.R tajimasD.R
theta.R TiTvRatio.R trimNs.R validIupacCodes.R variableSites.R
write.gtypes.R write.nexus.snapp.R gtypes2genind.R
gtypes2loci.R gtypes2phyDat.R df2gtypes.R sequence2gtypes.R

LazyData TRUE

VignetteBuilder knitr

LinkingTo Rcpp

RoxygenNote 6.0.1

NeedsCompilation yes

Author Eric Archer [aut, cre],
Paula Adams [aut],
Brita Schneiders [aut],
Sarina Fernandez [aut],
Warren Asfazadour [aut]

Maintainer Eric Archer <eric.archer@noaa.gov>

Repository CRAN

Date/Publication 2017-04-11 11:00:39 UTC

R topics documented:

`strataG-package` *Summaries and population structure analyses of DNA sequence genotypic data*

Description

strataG

`alleleFreqFormat` *Compiles and Formats Allele Frequencies*

Description

Format allele frequencies to for a set of ids and loci

Usage

```
alleleFreqFormat(x, g)
```

Arguments

x	a matrix or data.frame where first column is sample id and second colum is locus name.
g	a gtypes object.

Value

matrix of original samples, loci, and a formatted character string giving alleles and their overall frequencies in the data.

Author(s)

Eric Archer <eric.archer@noaa.gov>

See Also

alleleFreqs

Examples

```
data(msats.g)

x <- cbind(
 id = sample(indNames(msats.g), 10, rep = TRUE),
 locus = sample(locNames(msats.g), 10, rep = TRUE)
)
alleleFreqFormat(x, msats.g)
```

---

alleleFreqs *Allele Frequencies*

Description

Calculate allele frequencies for each locus.

Usage

```
alleleFreqs(g, by.strata = FALSE)
```

Arguments

g	a gtypes object.
by.strata	logical. If TRUE every element in the return list is a three dimensional array where the third dimension contains frequencies and proportions for each stratum.

Value

A list of allele frequencies for each locus. Each element is a matrix or array with frequencies by count (`freq`) and proportion (`prop`) of each allele.

Note

If g is a haploid object with sequences, make sure to run `labelHaplotypes` if sequences aren't already grouped by haplotype.

Author(s)

Eric Archer `<eric.archer@noaa.gov>`

See Also

alleleFreqFormat

Examples

```
data(msats.g)

f <- alleleFreqs(msats.g)
f$D11t # Frequencies and proportions for Locus D11t

f.pop <- alleleFreqs(msats.g, TRUE)
f.pop$EV94[, "freq", "Coastal"] # Frequencies for EV94 in the Coastal population
```

---

alleleSplit *Split Alleles For Diploid Data*

Description

Split loci stored in one column to two columns for each allele in a matrix of diploid data.

Usage

```
alleleSplit(x, sep = NULL)
```

Arguments

`x`	a matrix or data.frame containing diploid data. Every column represents one locus with two alleles.
`sep`	separator used between alleles of a locus. If `NULL`, then alleles should be of equal length (e.g., 145095 = 145 and 095, or AG = A and G).

Value

matrix with alleles for each locus in one column split into separate columns.

Author(s)

Eric Archer <eric.archer@noaa.gov>

Examples

```
# A sample SNP data set with no separators between nucleotides in a genotype
snps <- do.call(cbind, lapply(1:3, function(i) {
  a1 <- sample(c("A", "G"), 10, rep = TRUE)
  a2 <- sample(c("A", "G"), 10, rep = TRUE)
  paste(a1, a2, sep = "")
}))
colnames(snps) <- paste("Loc", LETTERS[1:3], sep = "_")
snps
alleleSplit(snps)


# A sample microsatellie data set with alleles separated by "/"
alleles <- seq(100, 150, 2)
msats <- do.call(cbind, lapply(1:3, function(i) {
  a1 <- sample(alleles, 10, rep = TRUE)
  a2 <- sample(alleles, 10, rep = TRUE)
  paste(a1, "/", a2, sep = "")
}))
colnames(msats) <- paste("Loc", LETTERS[1:3], sep = "_")
msats
alleleSplit(msats, sep = "/")
```

---

allelicRichness *Allelic Richness*

Description

Calculate allelic richness for each locus.

Usage

```
allelicRichness(g)
```

Arguments

g	a gtypes object.

Value

the allelic richness of each locus calculated as the number of alleles divided by the number of samples without missing data at that locus.

Author(s)

Eric Archer <eric.archer@noaa.gov>

Examples

```
data(msats.g)
allelicRichness(msats.g)
```

---

arlequin *Read and Write Arlequin Files*

Description

Read an Arlequin-formatted project input file into a gtypes object, or write an input file from a gtypes object.

Usage

```
read.arlequin(file)

write.arlequin(g, label = NULL, locus = 1)
```

Arguments

file	filename of an input arlequin project (.arp) file. read.arlequin can only read files with DataType of FREQUENCY, DNA, or MICROSAT.
g	a gtypes object.
label	label for Arlequin project filename (.arp). If NULL, the gtypes description is used if present.
locus	numeric or character designation of which locus to write for haploid data.

Author(s)

Eric Archer <eric.archer@noaa.gov>

References

Excoffier, L.G. Laval, and S. Schneider (2005) Arlequin ver. 3.0: An integrated software package for population genetics data analysis. Evolutionary Bioinformatics Online 1:47-50.
Available at http://cmpg.unibe.ch/software/arlequin3/

as.array.gtypes *Convert* gtypes *To* array

Description

Create a 3-dimensional array from a gtypes object.

Usage

```
## S4 method for signature 'gtypes'
as.array(x, ids = NULL, loci = NULL, drop = TRUE, ...)
```

Arguments

x	a gtypes object.
ids	vector of individual ids.
loci	vector of loci.
drop	if TRUE the array is coerced to the lowest possible dimension.
...	additional arguments to be passed to or from methods.

Value

A three dimensional array with with dimeinsions of:

Author(s)

Eric Archer <eric.archer@noaa.gov>

See Also

as.data.frame as.matrix

Examples

```
data(msats.g)
msats.arr <- as.array(msats.g)

msats.arr
```

`as.data.frame.gtypes` *Convert* `gtypes` *To* `data.frame`

Description

Create a data.frame from a gtypes object.

Usage

```
## S4 method for signature 'gtypes'
as.data.frame(x, one.col = FALSE, sep = "/",
  ids = TRUE, strata = TRUE, sort.alleles = TRUE, ...)
```

Arguments

`x`	a gtypes object.
`one.col`	logical. If `TRUE`, then result has one column per locus.
`sep`	character to use to separate alleles if `one.col` is `TRUE`.
`ids`	logical. include a column for individual identifiers (`ids`)?
`strata`	logical. include a column for current statification (`strata`)?
`sort.alleles`	logical. for non-haploid objects, should alleles be sorted in genotypes or left in original order? (only takes affect if `one.col = TRUE`)
`...`	additional arguments to be passed to or from methods.

Value

A `data.frame` with one row per sample.

Author(s)

Eric Archer `<eric.archer@noaa.gov>`

See Also

df2gtypes as.matrix as.array

Examples

```
data(msats.g)

# with defaults (alleles in multiple columns, with ids and stratification)
df <- as.data.frame(msats.g)
str(df)

# one column per locus
onecol.df <- as.data.frame(msats.g, one.col = TRUE)
str(onecol.df)
```

```
# just the genotypes
genotypes.df <- as.data.frame(msats.g, one.col = TRUE, ids = FALSE, strata = FALSE)
str(genotypes.df)
```

`as.matrix,gtypes-method` *Convert* `gtypes` *To* `matrix`

Description

Create a matrix from a gtypes object.

Usage

```
## S4 method for signature 'gtypes'
as.matrix(x, one.col = FALSE, sep = "/", ids = TRUE,
  strata = TRUE, sort.alleles = TRUE, ...)
```

Arguments

`x`	a gtypes object.
`one.col`	logical. If `TRUE`, then result has one column per locus.
`sep`	character to use to separate alleles if `one.col` is `TRUE`.
`ids`	logical. include a column for individual identifiers (`ids`)?
`strata`	logical. include a column for current statification (`strata`)?
`sort.alleles`	logical. for non-haploid objects, should alleles be sorted in genotypes or left as in original order? (only takes affect if `one.col = TRUE`)
`...`	additional arguments to be passed to or from methods.

Value

A `matrix` with one row per sample.

Author(s)

Eric Archer <eric.archer@noa.gov>

See Also

df2gtypes as.data.frame as.array

Examples

```
data(msats.g)

# with defaults (alleles in multiple columns, with ids and stratification)
mat <- as.matrix(msats.g)
head(mat)

# one column per locus
onecol.mat <- as.matrix(msats.g, one.col = TRUE)
head(onecol.mat)

# just the genotypes
genotypes.mat <- as.matrix(msats.g, one.col = TRUE, ids = FALSE, strata = FALSE)
head(genotypes.mat)
```

---

as.multidna *Convert to multidna*

Description

Convert a set of sequences to a multidna object if possible.

Usage

```
as.multidna(x)
```

Arguments

x	a valid set of sequences: character matrix, list of character vectors, DNAbin object or list of them, gtypes object, or multidna object.

Author(s)

Eric Archer <eric.archer@noaa.gov>

See Also

sequences, getSequences

Examples

```
# convert list of character vectors
data(dolph.seqs)
list.mdna <- as.multidna(dolph.seqs)
list.mdna

# convert gtypes object
data(dloop.g)
```

```
gtype.mdna <- as.multidna(dloop.g)
gtype.mdna
```

---

baseFreqs *Base Frequencies*

Description

Calculate nucleotide base frequencies along a sequence.

Usage

```
baseFreqs(x, bases = NULL, ignore = c("n", "x", "-", "."))
```

Arguments

x	a gtypes object with aligned sequences or a list of aligned DNA sequences.
bases	character vector of bases. Must contain valid IUPAC codes. If NULL, will return summary of frequencies of observed bases.
ignore	a character vector of bases to ignore when calculating frequencies.

Value

For each gene, a list containing:

site.freqs	a matrix of base frequencies at each site.
base.freqs	a vector of overall base proportion composition.

Author(s)

Eric Archer <eric.archer@noaa.gov>

Examples

```
data(dloop.g)
bf <- baseFreqs(dloop.g)

# Frequencies of first 10 sites
bf$site.freqs[, 1:10]

# Base composition
bf$base.freqs
```

bowhead.snp.position *Bowhead Whale SNP Genotype Groups*

Description

A data.frame of position information for SNPs to be phased

Usage

```
data(bowhead.snp.position)
```

Format

data.frame

References

Morin, P.A., Archer, F.I., Pease, V.L., Hancock-Hanser, B.L., Robertson, K.M., Huebinger, R.M., Martien, K.K., Bickham, J.W., George, J.C., Postma, L.D., Taylor, B.L., 2012. Empirical comparison of single nucleotide polymorphisms and microsatellites for population and demographic analyses of bowhead whales. Endangered Species Research 19, 129-147.

bowhead.snps *Bowhead Whale SNP Genotypes*

Description

A data.frame of 42 SNPs with sample ids and stratification

Usage

```
data(bowhead.snps)
```

Format

data.frame

References

Morin, P.A., Archer, F.I., Pease, V.L., Hancock-Hanser, B.L., Robertson, K.M., Huebinger, R.M., Martien, K.K., Bickham, J.W., George, J.C., Postma, L.D., Taylor, B.L., 2012. Empirical comparison of single nucleotide polymorphisms and microsatellites for population and demographic analyses of bowhead whales. Endangered Species Research 19, 129-147.

clumpp *Run CLUMPP*

Description

Run CLUMPP to aggregate multiple STRUCTURE runs.

Usage

```
clumpp(sr, k, align.algorithm = "greedy", sim.stat = "g",
  greedy.option = "ran.order", repeats = 100, order.by.run = 0,
  label = NULL, delete.files = TRUE)
```

Arguments

sr	result from structure or folder name containing STRUCTURE output files.
k	choice of *k* in sr to combine.
align.algorithm	algorithm to be used for aligning the runs. Can be "full.search", "greedy", or "large.k".
sim.stat	pairwise matrix similarity statistic to be used. Can be "g" or "g.prime".
greedy.option	input order of runs to be tested. Required if align.algorithm is "greedy" or "large.k". Valid choices are:

all	test all possible input orders of runs (note that this option increases the run-time sub-stantially unless R is small)
ran.order	test a specified number of random input orders of runs set by the repeats parameter.

repeats	the number of input orders of runs to be tested. Only used if align.algorithm is "greedy" or "large.k", and greedy.option is "ran.order".
order.by.run	permute the clusters according to the cluster order of a specific run. Set this parameter to a number from 1 to the number of runs in sr.
label	label to use for input and output files.
delete.files	logical. Delete all files when CLUMPP is finished?

Note

CLUMPP is not included with strataG and must be downloaded separately. Additionally, it must be installed such that it can be run from the command line in the current working directory. See the vignette for external.programs.

Author(s)

Eric Archer <eric.archer@noaa.gov>

References

Mattias Jakobsson and Noah A. Rosenberg. 2007. CLUMPP: a cluster matching and permutation program for dealing with label switching and multimodality in analysis of population structure. Bioinformatics 23(14):1801-1806. Available at http://web.stanford.edu/group/rosenberglab/clumppDownload.html

See Also

structure

`createConsensus` *Consensus Sequence*

Description

Return a consensus sequence from set of aligned sequences, introducing IUPAC ambiguity codes where necessary.

Usage

```
createConsensus(x, ignore.gaps = FALSE)
```

Arguments

`x`	a gtypes object with aligned sequences or a list of aligned DNA sequences.
`ignore.gaps`	logical. Ignore gaps at a site when creating consensus. If `TRUE`, then bases with a gap are removed before consensus is calculated. If `FALSE` and a gap is present, then the result is a gap.

Value

A character vector of the consensus sequence.

Author(s)

Eric Archer <eric.archer@noaa.gov>

Examples

```
data(dolph.seqs)
createConsensus(dolph.seqs)
```

`df2gtypes` *Convert a data.frame to gtypes*

Description

Load allelic data from a data.frame or matrix into a gtypes object.

Usage

```
df2gtypes(x, ploidy, id.col = 1, strata.col = 2, loc.col = 3,
  sequences = NULL, schemes = NULL, description = NULL, other = NULL)
```

Arguments

`x`	a matrix or data.frame of genetic data.
`ploidy`	number of number of columns in `x` storing alleles at each locus.
`id.col`	column name or number where individual sample ids are stored. If `NULL` then rownames are used. If there are no rownames, then samples are labelled with consecutive numbers.
`strata.col`	column name or number where stratification is stored. If `NULL` then all samples are in one (default) stratum.
`loc.col`	column number of first allele of first locus.
`sequences`	a list, matrix, `DNAbin`, or multidna object containing sequences.
`schemes`	an optional data.frame of stratification schemes.
`description`	a label for the object (optional).
`other`	a slot to carry other related information - unused in package analyses (optional).

Details

The genetic data in `x` starting at `loc.col` should be formatted such that every consecutive `ploidy` columns represent alleles of one locus. Locus names are taken from the column names in `x` and should be formatted with the same root locus name, with unique suffixes representing allels (e.g., for Locus1234: Locus1234.1 and Locus1234.2, or Locus1234_A and Locus1234_B).

If sequences are provided in `sequences`, then they should be named and match haplotype labels in `loc.col` of `x`. If multiple genes are given as a multidna, then they should have the same names as column names in `x` from `loc.col` to the end.

Value

a gtypes object.

Author(s)

Eric Archer <eric.archer@noaa.gov>

See Also

initialize.gtypes, sequence2gtypes, as.data.frame.gtypes, gtypes2genind, gtypes2loci

Examples

```
#--- create a diploid (microsatellite) gtypes object
data(dolph.msats)
ms.g <- df2gtypes(dolph.msats, ploidy = 2, strata.col = NULL, loc.col = 2)
ms.g

#' #--- create a haploid sequence (mtDNA) gtypes object
data(dolph.strata)
data(dolph.haps)

seq.df <- dolph.strata[ c("id", "broad", "dLoop")]
dl.g <- df2gtypes(seq.df, ploidy = 1, sequences = dolph.haps)
dl.g
```

---

dloop.g *Dolphin dLoop gtypes Object*

Description

A gtypes object of 126 samples and 33 haplotypes.

Usage

```
data(dloop.g)
```

Format

gtypes

References

Lowther-Thieleking J.L., F.I. Archer, A.R. Lang, and D.W. Weller. 2015. Genetic variation of coastal and offshore bottlenose dolphins, Tursiops truncatus, in the eastern North Pacific Ocean. Marine Mammal Science 31:1-20

dolph.haps *Dolphin mtDNA Haplotype Sequences*

Description

A list of 33 aligned d-loop haplotypes

Usage

```
data(dolph.haps)
```

Format

list

References

Lowther-Thieleking J.L., F.I. Archer, A.R. Lang, and D.W. Weller. 2015. Genetic variation of coastal and offshore bottlenose dolphins, Tursiops truncatus, in the eastern North Pacific Ocean. Marine Mammal Science 31:1-20

dolph.msats *Dolphin Microsatellite Genotypes*

Description

A data.frame of 126 samples and 4 microsatellite loci

Usage

```
data(dolph.msats)
```

Format

data.frame

References

Lowther-Thieleking J.L., F.I. Archer, A.R. Lang, and D.W. Weller. 2015. Genetic variation of coastal and offshore bottlenose dolphins, Tursiops truncatus, in the eastern North Pacific Ocean. Marine Mammal Science 31:1-20

dolph.seqs *Dolphin mtDNA D-loop Sequences*

Description

A list of 126 aligned control region sequences

Usage

```
data(dolph.seqs)
```

Format

list

References

Lowther-Thieleking J.L., F.I. Archer, A.R. Lang, and D.W. Weller. 2015. Genetic variation of coastal and offshore bottlenose dolphins, Tursiops truncatus, in the eastern North Pacific Ocean. Marine Mammal Science 31:1-20

dolph.strata *Dolphin Genetic Stratification and Haplotypes*

Description

A data.frame of 126 samples with assignment of samples to either broad-scale or fine-scale stratifications and mtDNA haplotype designations

Usage

```
data(dolph.strata)
```

Format

data.frame

References

Lowther-Thieleking J.L., F.I. Archer, A.R. Lang, and D.W. Weller. 2015. Genetic variation of coastal and offshore bottlenose dolphins, Tursiops truncatus, in the eastern North Pacific Ocean. Marine Mammal Science 31:1-20

dupGenotypes *Duplicate Genotypes*

Description

Identify duplicate or very similar genotypes.

Usage

```
dupGenotypes(g, num.shared = 0.8, num.cores = 1)
```

Arguments

g	a gtypes object.
num.shared	either number of loci or percentage of loci two individuals must share to be considered duplicate individuals.
num.cores	number of CPU cores to use.

Value

if no duplicates are present, the result is NULL, otherwise a data.frame with the following columns is returned:

ids.1, ids.2	sample ids.
strata.1, strata.2	sample stratification.
num.loci.genotyped	number of loci genotyped for both samples.
num.loci.shared	number of loci shared (all alleles the same) between both samples.
prop.loci.shared	proportion of loci genotyped for both samples that are shared.
mismatch.loci	loci where the two samples do not match.

Author(s)

Eric Archer <eric.archer@noaa.gov>

Examples

```
data(msats.g)

# identify potential duplicates in Coastal strata
dupes <- dupGenotypes(msats.g[, , "Coastal"])
dupes
```

evanno *Run Evanno Method on STRUCTURE Results*

Description

Calculate first and second order rates of changes of LnPr(K) from STRUCTURE results based on Evanno et al. 2005.

Usage

```
evanno(sr, plot = TRUE)
```

Arguments

sr	output from a call to structure.
plot	logical. Generate a plot of Evanno metrics?

Value

a list with:

df data.frame with Evanno log-likelihood metrics for each value of K.

plots list of four ggplot objects for later plotting.

Author(s)

Eric Archer <eric.archer@noaa.gov>

References

Evanno, G., Regnaut, S., and J. Goudet. 2005. Detecting the number of clusters of individuals using the software STRUCTURE: a simulation study. Molecular Ecology 14:2611-2620.

See Also

structure clumpp

Examples

```
## Not run:
data(msats.g)

# Run STRUCTURE
sr <- structureRun(msats, k.range = 1:4, num.k.rep = 10)

# Calculate Evanno metrics
evno <- evanno(sr)
evno
```

```
## End(Not run)
```

`expandHaplotypes` *Expand haplotypes*

Description

Expand haplotypes to a single sequence per individual.

Usage

```
expandHaplotypes(g)
```

Arguments

g	a haploid gtypes object with sequences.

Value

a gtypes object with sequences expanded and renamed so there is one sequence per individual. Sequence names are set to individual sample IDs.

Author(s)

Eric Archer <eric.archer@noaa.gov>

Examples

```
data(dloop.g)

# Haplotypes have already been labelled
dloop.g

# Haplotypes expanded to individual sequences (num.alleles == num.samples)
expanded.g <- expandHaplotypes(dloop.g)
expanded.g
```

fasta *Read and Write FASTA*

Description

Read and write FASTA formatted files of sequences.

Usage

```
read.fasta(file)

write.fasta(x, file = "sequences.fasta")
```

Arguments

file	a FASTA-formatted file of sequences.
x	a list or a matrix of DNA sequences (see write.dna), or a gtypes object with sequences.

Value

read.fasta a set of sequences in DNAbin format

write.fasta invisbly, name(s) of file(s) written

Author(s)

Eric Archer <eric.archer@noaa.gov>

fastsimcoal *Run fastsimcoal*

Description

Run a fastsimcoal simulation and load results into a gtypes object.

Usage

```
fscWrite(pop.info, locus.params, mig.rates = NULL, hist.ev = NULL,
  label = NULL)

fscRead(file, locus.params, label.haplotypes = TRUE)

fastsimcoal(pop.info, locus.params, mig.rates = NULL, hist.ev = NULL,
  label = NULL, quiet = TRUE, delete.files = TRUE, exec = "fsc252",
  num.cores = NULL, label.haplotypes = TRUE)
```

Arguments

pop.info	matrix of population sampling information created by the fscPopInfo function.
locus.params	data.frame specifying loci to simulate created by the fscLocusParams function.
mig.rates	a matrix or list of matrices giving the migration rates between pairs of populations.
hist.ev	matrix of historical events created by the fscHistEv function.
label	character string to label files with.
file	filename to write to.
label.haplotypes	if DNA sequences are being simulated, should resulting sequences be stored as haplotypes (default = TRUE), or left as individual sequences (FALSE)?
quiet	logical. Run quietly?
delete.files	logical. Delete files when done?
exec	name of fastsimcoal executable.
num.cores	number of cores to use.

Note

fastsimcoal is not included with strataG and must be downloaded separately. Additionally, it must be installed such that it can be run from the command line in the current working directory. See the vignette for external.programs for installation instructions.

Author(s)

Eric Archer <eric.archer@noaa.gov>

References

Excoffier, L. and Foll, M (2011) fastsimcoal: a continuous-time coalescent simulator of genomic diversity under arbitrarily complex evolutionary scenarios Bioinformatics 27: 1332-1334.
http://cmpg.unibe.ch/software/fastsimcoal2/

Examples

```
## Not run:
# Set fastsimcoal parameters
# Population information: 3 populations with Ne = 10,000, drawing 100 samples from each.
pop.info <- fscPopInfo(pop.size = rep(10000, 3), sample.size = rep(100, 3))

# Migration rates among the 3 populations
mig.rates <- matrix(c(0, 0.5, 0.005, 0.5, 0, 0.0005, 0.005, 0.0005, 0), ncol = 3)

# Define historical events in which populations diverged 2000 generations in past
hist.ev <- fscHistEv(
  num.gen = c(2000, 2000), source.deme = c(2, 1),
  sink.deme = c(1, 0), prop.migrants = 1
)
```

```
# Define 10 microsatellite loci, with random mutation rates
msat.params <- fscLocusParams(
  locus.type = "msat", num.loci = 1,
  mut.rate = runif(10, 1e-7, 1e-4)
)

# Run simulation and display locus summary
sim.msats <- fastsimcoal(pop.info, msat.params, mig.rates, hist.ev)
summarizeLoci(sim.msats)

## End(Not run)
```

`fastsimcoal.input` *Input functions for fastsimcoal parameters*

Description

Functions to create `pop.info`, `locus.params`, and `hist.ev` input matrices for fastsimcoal function.

Usage

```
fscPopInfo(pop.size, sample.size, sample.times = 0, growth.rate = 0)

fscLocusParams(locus.type = c("dna", "msat", "snp"), sequence.length = NULL,
  num.loci = NULL, mut.rate = NULL, transition.rate = 1/3,
  gsm.param = 0, range.constraint = 0, recomb.rate = 0,
  chromosome = NULL, num.chrom = NULL, ploidy = 2)

fscHistEv(num.gen = 0, source.deme = 0, sink.deme = 0,
  prop.migrants = 1, new.sink.size = 1, new.sink.growth = 0,
  new.mig.mat = 0)
```

Arguments

`pop.size`	a vector giving size of each populaiton.
`sample.size`	a vector giving the number of samples to take from each population.
`sample.times`	a vector giving the number of generations in the past at which samples are taken.
`growth.rate`	a vector giving the growth rate of each population.
`locus.type`	a character representation of what type of marker to simulate. Can be "dna", "msat", or "snp".
`sequence.length`	`dna`: number of DNA base pairs to use.
`num.loci`	`msat, snp`: number of loci to simulate.
`mut.rate`	`dna, msat`: per base pair or locus mutation rate.

`transition.rate`
: dna: fraction of substitutions that are transitions. Set to 1 (all transitions) for SNPs.

`gsm.param`
: `msat`: Value of the geometric parameter for a Generalized Stepwise Mutation (GSM) model. This value represents the proportion of mutations that will change the allele size by more than one step. Values between 0 and 1 are required. A value of 0 is for a strict Stepwise Mutation Model (SMM).

`range.constraint`
: `msat`: Range constraint (number of different alleles allowed). A value of 0 means no range constraint.

`recomb.rate`
: recombination rate between adjacent markers. No effect for SNPs.

`chromosome`
: number or character identifying which chromosome the marker is on.

`num.chrom`
: a value giving the number of chromosomes that the `locus.params` marker specifications should be copied for. If `NULL`, then chromosome assignment is taken from the `chromosome` column. Any non-`NULL` integer will cause the value in `chromosome`.

`ploidy`
: positive integer giving the ploidy of the marker type to be simulated.

`num.gen`
: Number of generations, t, before present at which the historical event happened.

`source.deme`
: Source deme (the first listed deme has index 0)

`sink.deme`
: Sink deme

`prop.migrants`
: Expected proportion of migrants to move from source to sink.

`new.sink.size`
: New size for the sink deme, relative to its size at generation t.

`new.sink.growth`
: New growth rate for the sink deme.

`new.mig.mat`
: New migration matrix to be used further back in time.

Note

SNPs are simulated as a diploid DNA sequence with a single nucleotide, with no recombination (each on its own "chromosome"). If `mut.rate` has more than one value then this many SNPs are simulated. Otherwise, `num.loci` independent SNPs are simulated.

Author(s)

Eric Archer `<eric.archer@noaa.gov>`

References

Excoffier, L. and Foll, M (2011) fastsimcoal: a continuous-time coalescent simulator of genomic diversity under arbitrarily complex evolutionary scenarios Bioinformatics 27: 1332-1334.
http://cmpg.unibe.ch/software/fastsimcoal2/

See Also

`fastsimcoal`

`fixedDifferences` *Fixed Differences*

Description

Summarize fixed base pair differences between strata.

Usage

```
fixedDifferences(g, count.indels = TRUE, consec.indels.as.one = TRUE,
  bases = c("a", "c", "g", "t", "-"))
```

Arguments

`g`	a gtypes object.
`count.indels`	logical. Count indels when evaluating sites for fixed differences?
`consec.indels.as.one`	logical. If `count.indels` is TRUE, count consecutive indels as a a single indel?
`bases`	a character vector of valid bases to consider.

Value

a list with components:

sites list of sites with fixed differences for each pair of strata

num.fixed data.frame of number of sites fixed between each pair of strata

Author(s)

Eric Archer <eric.archer@noaa.gov>

See Also

`fixedSites`, `variableSites`

Examples

```
data(dloop.g)
fd <- fixedDifferences(dloop.g)
fd
```

fixedSites *Fixed Sites*

Description

Identify fixed sites among sequences.

Usage

```
fixedSites(x, bases = c("a", "c", "g", "t", "-"))
```

Arguments

x	a gtypes object with sequences, a list of sequences, or a consensus sequence. Sequences must be aligned.
bases	a character vector of valid bases to consider.

Value

a vector of fixed sites. Element names are site positions in the original sequence.

Author(s)

Eric Archer <eric.archer@noaa.gov>

See Also

variableSites

Examples

```
data(dolph.haps)

fixedSites(dolph.haps)
```

freq2GenData *Convert Haplotype Frequency Matrices*

Description

Create a data.frame of stratified individuals and their haplotypes from a frequency table

Usage

```
freq2GenData(freq.mat, hap.col = NULL, freq.col = 1, id.label = NULL,
  hap.label = NULL)
```

Arguments

freq.mat	a matrix or data.frame containing haplotypic frequencies with strata as column names.
hap.col	a number giving the column providing haplotype labels or a vector the same length as freq.mat. If NULL rownames are used.
freq.col	a number giving the first column containing haplotype frequencies.
id.label	character to label sample IDs with in resulting data.frame.
hap.label	character to label haplotypes with in resulting data.frame.

Value

a data.frame with one row per sample and columns for id, strata, and haplotype, suitable for use in df2gtypes.

Author(s)

Eric Archer <eric.archer@noaa.gov>

Examples

```
hap.freqs <- data.frame(
  haps = c("hap1", "hap2", "hap3"),
  pop1 = rmultinom(1, 50, prob = c(0.1, 0.2, 0.7)),
  pop2 = rmultinom(1, 25, prob = c(0.5, 0.4, 0.1))
)

gen.data <- freq2GenData(hap.freqs, hap.col = 1, freq.col = 2)

x <- df2gtypes(gen.data, ploidy = 1)
summary(x)
```

---

fusFs *Fu's Fs*

Description

Calculate Fu's Fs for a set of sequences to test for selection.

Usage

```
fusFs(x)
```

Arguments

x	set of DNA sequences or a haploid gtypes object with sequences.

Note

Currently, this function is limited to calculating Fs for fewer than approximately 172 sequences due to numerical overflow issues. NaN will be returned for larger data sets.

Author(s)

Eric Archer <eric.archer@noaa.gov>

References

Fu, Y-X. 1997. Statistical tests of neutrality of mutations against population growth, hitchiking and background selection. Genetics 147:915-925.

Examples

```
data(dolph.seqs)

fusFs(dolph.seqs)
```

---

gelato *GELATo - Group ExcLusion and Assignment Test*

Description

Run a GELATo test to evaluate assignment likelihoods of groups of samples.

Usage

```
gelato(g, unknown.strata, nrep = 1000, min.sample.size = 5)

gelatoPlot(gelato.result, unknown = NULL, main = NULL)
```

Arguments

g	a gtypes object.
unknown.strata	a character vector listing to assign. Strata must occur in g.
nrep	number of permutation replicates for Fst distribution.
min.sample.size	minimum number of samples to use to characterize knowns. If the known sample size would be smaller than this after drawing an equivalent number of unknowns for self-assignment, then the comparison is not done.
gelato.result	the result of a call to gelato.
unknown	the names of the unknown strata in the x$likelihoods element to create plots. If NULL one plot for each stratum is created.
main	main label for top of plot.

Value

A list with the following elements:

assign.prob	a data.frame of assignment probabilities.
likelihoods	a list of likelihoods.

Author(s)

Eric Archer <eric.archer@noaa.gov>

References

O'Corry-Crowe, G., W. Lucey, F.I. Archer, and B. Mahoney. 2015. The genetic ecology and population origins of the beluga whales of Yakutat Bay. Marine Fisheries Review.

Examples

```
data(msats.g)

# Run GELATo analysis
gelato.fine <- gelato(msats.g, unk = "Offshore.South", nrep = 20)
gelato.fine

# Plot results
gelatoPlot(gelato.fine, "Offshore.South")
```

---

genepop *Run GENEPOP*

Description

Format output files and run GENEPOP. Filenames used are returned so that output files can be viewed or read and parsed into R.

Usage

```
genepop(g, output.ext = "", show.output = F, label = "genepop.run",
  dem = 10000, batches = 100, iter = 5000, other.settings = "",
  input.fname = "loc_data.txt", exec = "Genepop")

genepopWrite(g, label = "genepop.write", input.fname = "loc_data.txt")
```

Arguments

g	a gtypes object.
output.ext	character string to use as extension for output files.
show.output	logical. Show GENEPOP output on console?
label	character string to use to label GENEPOP input and output files.
dem	integer giving the number of MCMC dememorisation or burnin steps.
batches	integer giving number of MCMC batches.
iter	integer giving number of MCMC iterations.
other.settings	character string of optional GENEPOP command line arguments.
input.fname	character string to use for input file name.
exec	name of Genepop executable

Value

genepop a list with a vector of the locus names and a vector of the input and output filenames

genepopWrite a vector of the locus names used in the input file

Note

GENEPOP is not included with strataG and must be downloaded separately. Additionally, it must be installed such that it can be run from the command line in the current working directory. See the vignette for external.programs for installation instructions.

Author(s)

Eric Archer <eric.archer@noaa.gov>

References

GENEPOP 4.3 (08 July 2014; Rousset, 2008)
http://kimura.univ-montp2.fr/~rousset/Genepop.htm

See Also

hweTest, LDgenepop

Examples

```
## Not run:
# Estimate Nm for the microsatellite data
data(msats.g)
# Run Genepop for Option 4
results <- genepop(msats.g, output.ext = ".PRI", other.settings = "MenuOptions=4")
# Locus name mapping and files
results
# Show contents of output file
file.show(results$files["output.fname"])
```

```
## End(Not run)
```

---

gtypes-class gtypes *Class*

Description

An S4 class storing multi-allelic locus or sequence data along with a current stratification and option stratification schemes.

Slots

`data` a data.table where the first column contains the sample ID (`ids`). The second column contains the sample stratification (`strata`). The third column to the end contains the allelic data as one column per locus. Alleles are on multiple rows per column with sample IDs duplicated for all alleles. Column names are unique locus names.

`sequences` a multidna object.

`ploidy` integer representing the ploidy of the data. There are ploidy * the number of individuals rows in 'data'.

`schemes` a data.frame with stratification schemes in each column. The rownames are individual names and must match the 'id' column of the 'data' slot. Each column is a factor.

`description` a label for the object (optional).

`other` a slot to carry other related information - currently unused in analyses (optional).

Author(s)

Eric Archer <eric.archer@noaa.gov>

See Also

df2gtypes, sequence2gtypes, genind2gtypes, gtypes.accessors, initialize.gtypes

Examples

```
#--- create a diploid (microsatellite) gtypes object
data(dolph.msats)
data(dolph.strata)
strata.schemes <- dolph.strata[, c("broad", "fine")]
rownames(strata.schemes) <- dolph.strata$id
msats.g <- new("gtypes", gen.data = dolph.msats[, -1], ploidy = 2,
               ind.names = dolph.msats[, 1], schemes = strata.schemes)
msats.g

#--- create a haploid sequence (mtDNA) gtypes object and label haplotypes
```

```
data(dolph.seqs)
dloop.haps <- cbind(dLoop = dolph.strata$id)
rownames(dloop.haps) <- dolph.strata$id
dloop.g <- new("gtypes", gen.data = dloop.haps, ploidy = 1,
               schemes = strata.schemes, sequences = dolph.seqs,
               strata = "fine")
dloop.g
dloop.g <- labelHaplotypes(dloop.g, "Hap.")$gtypes
dloop.g
```

---

gtypes.accessors gtypes *Accessors*

Description

Accessors for slots in gtypes objects.

Usage

```
## S4 method for signature 'gtypes'
nInd(x, ...)

## S4 method for signature 'gtypes'
nLoc(x, ...)

nStrata(x, ...)

## S4 method for signature 'gtypes'
nStrata(x, ...)

## S4 method for signature 'gtypes'
indNames(x, ...)

## S4 method for signature 'gtypes'
locNames(x, ...)

strataNames(x, ...)

## S4 method for signature 'gtypes'
strataNames(x, ...)

## S4 method for signature 'gtypes'
ploidy(x, ...)

## S4 method for signature 'gtypes'
other(x, ...)
```

```
## S4 method for signature 'gtypes'
strata(x)

strata(x) <- value

## S4 replacement method for signature 'gtypes'
strata(x) <- value

schemes(x, ...)

## S4 method for signature 'gtypes'
schemes(x, ...)

schemes(x) <- value

## S4 replacement method for signature 'gtypes'
schemes(x) <- value

alleleNames(x, ...)

## S4 method for signature 'gtypes'
alleleNames(x)

sequences(x, ...)

## S4 method for signature 'gtypes'
sequences(x, seqName = NULL, as.haplotypes = TRUE, ...)

description(x, ...)

## S4 method for signature 'gtypes'
description(x, ...)

description(x) <- value

## S4 replacement method for signature 'gtypes'
description(x) <- value

## S4 method for signature 'gtypes,ANY,ANY,ANY'
x[i, j, k, ..., quiet = TRUE,
  drop = FALSE]
```

Arguments

x	a gtypes object.
...	other arguments passed from generics (ignored).
value	value being assigned by accessor.
seqName	the name (or number) of a set of sequences from the @sequences slot to return.

as.haplotypes	return sequences as haplotypes? If TRUE, contents of @sequences slot are returned. If FALSE, one sequence per individual is returned.
i, j, k	subsetting slots for individuals (i), loci (j), or strata (k). See Details for more information.
quiet	suppress warnings about unmatched requested individuals, loci, or strata?
drop	if TRUE the return object will have unused sequences removed.

Details

Indexing a gtypes object with integers, characters, or logicals with the [operator follows the same rules as normal indexing in R. The order that individuals, loci, and strata are chosen in the order returned by indNames, locNames, and strataNames respectively. If unstratified samples are present, they can be selected as a group either by including NA in the character or numeric vector of the k slot, or by providing a logical vector based on is.na(strata(g)) to the i slot.

Value

nInd number of individuals

nLoc number of loci

nStrata number of strata

indNames vector of individual/sample names

locNames vector of locus names

strataNames vector of strata names for current scheme

ploidy number of alleles at each locus

other contents of @other slot

strata return or modify the current stratification

schemes return or modify the current stratification schemes

alleleNames return a list of alleles at each locus

sequences return the multidna object in the @sequences slot. See getSequences to extract individual genes or sequences from this object

description return the object's description

Author(s)

Eric Archer <eric.archer@noaa.gov>

Examples

```
#--- create a diploid (microsatellite) gtypes object
data(msats.g)
msats.g <- stratify(msats.g, "fine")

nStrata(msats.g)
strataNames(msats.g)
nLoc(msats.g)
```

```
locNames(msats.g)

# reassign all samples to two randomly chosen strata
strata(msats.g) <- sample(c("A", "B"), nInd(msats.g), rep = TRUE)
msats.g


#--- a sequence example
library(ape)
data(woodmouse)
genes <- list(gene1=woodmouse[,1:500], gene2=woodmouse[,501:965])
x <- new("multidna", genes)
wood.g <- sequence2gtypes(x)
strata(wood.g) <- sample(c("A", "B"), nInd(wood.g), rep = TRUE)
wood.g

# get the multidna sequence object
multi.seqs <- sequences(wood.g)
class(multi.seqs) # "multidna"

# get a list of DNAbin objects
library(apex)
dnabin.list <- getSequences(multi.seqs)
class(dnabin.list) # "list"

# get a DNAbin object of the first locus
dnabin.1 <- getSequences(multi.seqs, locNames(wood.g)[1])
class(dnabin.1) # "DNAbin"

# NOTE: The default to the 'simplify' argument in 'getSequences' is TRUE,
#   so if there is only one locus, 'getSequences' will return a DNAbin object
#   rather than a single element list unless 'simplify = FALSE':
gene1 <- wood.g[, "gene1", ]
gene1.dnabin <- getSequences(sequences(gene1))
class(gene1.dnabin) # "DNAbin"
```

---

gtypes2genind *Convert Between* gtypes *And* genind *objects.*

Description

Convert a gtypes object to a genind object and vice-versa.

Usage

```
gtypes2genind(x, type = c("codom", "PA"))

genind2gtypes(x)
```

Arguments

x	either a gtypes or genind object to convert from.
type	a character string indicating the type of marker for genind objects: 'codom' stands for 'codominant' (e.g. microstallites, allozymes); 'PA' stands for 'presence/absence' markers (e.g. AFLP, RAPD).

Author(s)

Eric Archer <eric.archer@noaa.gov>

See Also

initialize.gtypes, df2gtypes, sequence2gtypes, as.data.frame.gtypes, gtypes2loci

Examples

```
data(msats.g)

# Convert to genind
gi <- gtypes2genind(msats.g)
gi

# Convert to gtypes
gt <- genind2gtypes(gi)
gt
```

---

gtypes2loci *Convert Between* gtypes *And* loci *objects.*

Description

Convert a gtypes object to a loci object.

Usage

```
gtypes2loci(x)

loci2gtypes(x, description = NULL)
```

Arguments

x	a gtypes or loci formatted object.
description	a label for the gtypes object (optional).

Author(s)

Eric Archer <eric.archer@noaa.gov>

See Also

initialize.gtypes, df2gtypes, sequence2gtypes, as.data.frame.gtypes, gtypes2genind

Examples

```
data(msats.g)

# Convert to loci
lc <- gtypes2loci(msats.g)
lc

# Convert to gtypes
gt <- loci2gtypes(lc)
gt
```

`gtypes2phyDat` *Convert Between* `gtypes` *And* `phyDat` *objects.*

Description

Convert a `gtypes` object to a `phyDat` object.

Usage

```
gtypes2phyDat(x, locus = 1)

phyDat2gtypes(x, ...)
```

Arguments

`x`	a gtypes or `phyDat` formatted object.
`locus`	name or number of locus to convert.
`...`	optional arguments passed to `sequence2gtypes`.

Author(s)

Eric Archer `<eric.archer@noaa.gov>`

See Also

initialize.gtypes, df2gtypes, sequence2gtypes, as.data.frame.gtypes, gtypes2genind, gtypes2loci

Examples

```
data(dloop.g)

# Convert to phDat
pd <- gtypes2phyDat(dloop.g)
pd

# Convert to gtypes
gt <- phyDat2gtypes(pd)
gt
```

`heterozygosity` *Observed and Expected Heterozygosity*

Description

Calculate observed heterozygosity for diploid data.

Usage

```
exptdHet(g)

obsvdHet(g)
```

Arguments

g	a gtypes object.

Note

For a measure of haplotypic diversity (haploid "heterozygosity"), use `exptdHet`. If `g` is a haploid object with sequences, make sure to run `labelHaplotypes` if sequences aren't already grouped by haplotype.

Author(s)

Eric Archer <eric.archer@noaa.gov>

Examples

```
data(msats.g)

# Expected heterozygosity
exptdHet(msats.g)

# Observed heterozygosity
obsvdHet(msats.g)
```

hweTest *Hardy-Weinberg Equilibrium*

Description

Calculate Hardy-Weinberg equilibrium p-values.

Usage

```
hweTest(g, use.genepop = FALSE, B = 1000, show.output = FALSE,
  delete.files = TRUE, label = "HWE.genepop", ...)
```

Arguments

g	a gtypes object.
use.genepop	logical. Use GENEPOP to calculate HWE p-values? If FALSE then hw.test is used.
B	the number of replicates for the Monte Carlo procedure for hw.test.
show.output	logical. Show output from GENEPOP?
delete.files	logical. Delete GENEPOP files when done?
label	character string to use to label GENEPOP files.
...	arguments to be passed to genepop.

Value

a vector of p-values for each locus.

Note

If use.genepop = TRUE, the command line version of GENEPOP v.4 must be properly installed and available on the command line, so it is executable from any directory. On PC's, this requires having it in a folder in the PATH environmental variable. On Macs, the executable should be installed in a folder like /usr/local/bin.

Author(s)

Eric Archer <eric.archer@noaa.gov>

See Also

genepop, hw.test

Examples

```
data(msats.g)
hweTest(msats.g)
```

initialize,gtypes-method
gtypes *Constructor*

Description

Create a new gtypes object using `new("gtypes", ...)`, where '`...`' are arguments documented below.

Usage

```
## S4 method for signature 'gtypes'
initialize(.Object, gen.data, ploidy, ind.names = NULL,
  sequences = NULL, strata = NULL, schemes = NULL, description = NULL,
  other = NULL, remove.sequences = FALSE)
```

Arguments

`.Object`	the object skeleton, automatically generated when calling `new`.
`gen.data`	a vector, matrix, or data.frame containing the alleles at each locus. See below for more details.
`ploidy`	ploidy of the loci.
`ind.names`	an optional vector of individual sample names.
`sequences`	an optional multidna object containing sequences represented by each locus.
`strata`	an optional stratification scheme from `schemes`.
`schemes`	an optional data.frame of stratification schemes.
`description`	an optional description for the object.
`other`	other optional information to include.
`remove.sequences`	logical. If `TRUE` any sequences not referenced

Details

For multi-allele loci, the `gen.data` argument should be formatted such that every consecutive `ploidy` columns represent alleles of one locus. Locus names are taken from the column names in `gen.data` and should be formatted with the same root locus name, with unique suffixes representing alleles (e.g., for Locus1234: Locus1234.1 and Locus1234.2, or Locus1234_A and Locus1234_B).

If `gen.data` is a vector it is assumed to represent haplotypes of a haploid marker. Sample names can be either in the rownames of `gen.data` or given separately in `ind.names`. If `ind.names` are provided, these are used in lieu of rownames in `gen.data` and also used to label `schemes`. If sequences are provided in `sequences`, then they should be named and match haplotype labels in `loc.col` of `x`. If multiple genes are given as a multidna, then they should have the same names as column names in `X` from `loc.col` to the end.

Author(s)

Eric Archer <eric.archer@noaa.gov>

See Also

df2gtypes, sequence2gtypes, gtypes2genind, gtypes2loci

`is.gtypes` *Test if object is* `gtypes`

Description

Test if object is `gtypes`

Usage

```
is.gtypes(x)
```

Arguments

`x`	R object to be tested.

Value

Logical stating if 'x' is a `gtypes` object.

Author(s)

Eric Archer <eric.archer@noaa.gov>

Examples

```
data(msats.g)
is.gtypes(msats.g) # TRUE

data(dolph.msats)
is.gtypes(dolph.msats) # FALSE
```

iupacCode *IUPAC Code*

Description

Calculate the correct IUPAC code for a vector of nucleotides.

Usage

```
iupacCode(bases, ignore.gaps = FALSE)
```

Arguments

bases	character vector containing valid nucleotides or IUPAC codes.
ignore.gaps	logical. Ignore gaps at a site when creating consensus. If true, then bases with a gap are removed before consensus is calculated. If false and a gap is present, then the result is a gap.

Value

a character representing the correct IUPAC code.

Author(s)

Eric Archer <eric.archer@noaa.gov>

See Also

validIupacCodes

Examples

```
iupacCode(c("a", "a", "g"))

iupacCode(c("t", "c", "g"))
```

jackHWE *Hardy-Weinberg Equlibrium Jackknife*

Description

Test influence of samples on Hardy-Weinberg equilibrium via jackknife.

Usage

```
jackHWE(g, exclude.num = 1, min.hwe.samples = 5, show.progress = TRUE,
  use.genepop = FALSE, ...)

jackInfluential(jack.result, alpha = 0.05)

## S3 method for class 'jack.influential'
plot(x, main = NULL, ...)
```

Arguments

g	a gtypes object.
exclude.num	Number of samples to exclude at a time.
min.hwe.samples	minimum samples needed to calculate HWE.
show.progress	logical. Show progress of jackknife?
use.genepop	logical. Use GENEPOP to calculate HWE p-values? If FALSE then hw.test is used.
...	other arguments to be passed to GENEPOP.
jack.result	result from run of jackHWE.
alpha	critical value to determine if exclusion is "influential".
x	result from a call to jackInfluential.
main	main title for influential sample plots from plot.jack.influential.

Details

jackHWE performs a HWE jackknife where all combinations of exclude.num samples are left out and HWE is recalculated

jackInfluential calculates odds.ratios between jackknife HWE and observed HWE and identifies "influential" samples. Samples are "influential" if the observed HWE p-value is < alpha, but is > alpha when the samples are not present

plot.jack.influential creates a cumulative frequency plot of all odds-ratios from jack.influential. A vertical dashed line marks the smallest influential exclusion

Value

`jackHWE` returns a list with:

`obs`	a named vector of HWE p-values for each locus.
`jack`	a `data.frame` of HWE p-values where each row is an exclusion and columns are loci.
`gtypes`	the original `gtypes` object.

`jackInfluential` returns a list with:

`influential`	a `data.frame` of influential exclusions.
`allele.freqs`	a `data.frame` listing the allele frequencies of influential exclusions.
`odds.ratio`	a `matrix` of odds ratios between exclusions (rows) and loci (columns).

Note

If `use.genepop = TRUE`, the command line version of GENEPOP v.4 must be properly installed and available on the command line, so it is executable from any directory. On PC's, this requires having it in a folder in the PATH environmental variable. On Macs, the executable should be installed in a folder like `/usr/local/bin`.

Author(s)

Eric Archer `<eric.archer@noaa.gov>`

References

Morin, P.A., R.G. LeDuc, F.I. Archer, K.K. Martien, R. Huebinger, J.W. Bickham, and B.L. Taylor. 2009. Significant deviations from Hardy-Weinberg equilibirum caused by low levels of microsatellite genotyping errors. Molecular Ecology Resources 9:498-504.

See Also

`hweTest`

`labelHaplotypes` *Find and label haplotypes*

Description

Identify and group sequences that share the same haplotype.

Usage

```
labelHaplotypes(x, prefix = NULL, use.indels = TRUE)

## Default S3 method:
labelHaplotypes(x, prefix = NULL, use.indels = TRUE)

## S3 method for class 'list'
labelHaplotypes(x, ...)

## S3 method for class 'character'
labelHaplotypes(x, ...)

## S3 method for class 'gtypes'
labelHaplotypes(x, ...)
```

Arguments

x	sequences in a character matrix, list, or DNAbin object, or a haploid gtypes object with sequences.
prefix	a character string giving prefix to be applied to numbered haplotypes. If NULL, haplotypes will be labeled with the first label from original sequences.
use.indels	logical. Use indels when comparing sequences?
...	arguments to be passed to labelHaplotypes.default.

Details

If any sequences contain ambiguous bases (N's) they are first removed. Then haplotypes are assigned based on the remaining sequences. The sequences with N's that were removed are then assigned to the new haplotypes if it can be done unambiguously (they match only one haplotype with 0 differences once the N's have been removed). If this can't be done they are assigned NAs and listed in the unassigned element.

Value

For character, list, or DNAbin, a list with the following elements:

haps named vector (DNAbin) or list of named vectors (multidina) of haplotypes for each sequence in x.

hap.seqs DNAbin or multidna object containing sequences for each haplotype.

unassigned data.frame listing closest matching haplotypes for unassignable sequences with N's and the minimum number of substitutions between the two. Will be NULL if no sequences remain unassigned.

For gtypes, a list with the following elements:

gtypes the new gtypes object with the haplotypes reassigned.

unassigned a list containing the unassigned data.frame for each gene if present, otherwise NULL.

Author(s)

Eric Archer <eric.archer@noaa.gov>

Examples

```
# create 5 example short haplotypes
haps <- c(
  H1 = "ggctagct",
  H2 = "agttagct",
  H3 = "agctggct",
  H4 = "agctggct",
  H5 = "ggttagct"
)
# draw and label 100 samples
sample.seqs <- sample(names(haps), 100, rep = TRUE)
ids <- paste(sample.seqs, 1:length(sample.seqs), sep = "_")
sample.seqs <- lapply(sample.seqs, function(x) strsplit(haps[x], "")[[1]])
names(sample.seqs) <- ids

# add 1-2 random ambiguities
with.error <- sample(1:length(sample.seqs), 10)
for(i in with.error) {
  num.errors <- sample(1:2, 1)
  sites <- sample(1:length(sample.seqs[[i]]), num.errors)
  sample.seqs[[i]][sites] <- "n"
}

hap.assign <- labelHaplotypes(sample.seqs, prefix = "Hap.")
hap.assign
```

LDgenepop *Linkage Disequlibrium*

Description

Calculate linkage disequilibrium p-values using GENEPOP.

Usage

```
LDgenepop(g, show.output = FALSE, delete.files = TRUE,
  label = "linkage.genepop", ...)
```

Arguments

g	a gtypes object.
show.output	logical. Show GENEPOP output on console?
delete.files	logical. Delete GENEPOP input and output files when done?

label	character string to use to label GENEPOP input and output files.
...	other arguments to be passed to genepop.

Value

data.frame of disequilibrium estimates between pairs of individuals

Author(s)

Eric Archer <eric.archer@noaa.gov>

See Also

genepop

Examples

```
## Not run:
data(msats.g)
msats.ld <- LDgenepop(msats.g)
head(msats.ld)

## End(Not run)
```

---

ldNe *ldNe*

Description

Estimate Ne from linkage disequilibrium based on Pearson correlation approximation following Waples et al 2016. Adapted from code by R. Waples and W. Larson.

Usage

```
ldNe(g, maf.threshold = 0, by.strata = FALSE, ci = 0.95)
```

Arguments

g	a gtypes object.
maf.threshold	smallest minimum allele frequency permitted to include a locus in calculation of Ne.
by.strata	apply the maf.threshold by strata. If TRUE then any locus that is below this threshold in any strata will be removed from the calculation of Ne for every stratum. Otherwise, loci are removed only if they are below the maf.threshold in the stratum for which Ne is calculated.
ci	central confidence interval.

Value

a numeric matrix with one row per strata and the following columns:

`S` harmonic mean of sample size across pairwise comparisons of loci

`num.comp` number of pairwise loci comparisons used

`mean.rsq` mean r^2 over all loci

`mean.E.rsq` mean expected r^2 over all loci

`Ne` estimated Ne

`param.lci, param.uci` parametric lower and upper CIs

Author(s)

Eric Archer `<eric.archer@noaa.gov>`

References

Waples, R.S. 2006. A bias correction for estimates of effective population size based on linkage disequilibrium at unlinked gene loci. Conservation Genetics 7:167-184.

Waples RK, Larson WA, and Waples RS. 2016. Estimating contemporary effective population size in non-model species using linkage disequilibrium across thousands of loci. Heredity 117:233-240; doi:10.1038/hdy.2016.60

---

lowFreqSubs *Low Frequency Substitutions*

Description

Check nucleotide sites for low frequency substitutions.

Usage

```
lowFreqSubs(x, min.freq = 3, motif.length = 10, ...)
```

Arguments

`x`	a `DNAbin` object.
`min.freq`	minimum frequency of base to be flagged.
`motif.length`	length of motif around low frequency base to output.
`...`	arguments passed from other functions (ignored).

Value

data.frame listing id, site number, and motif around low frequency base call.

Author(s)

Eric Archer <eric.archer@noaa.gov>

Examples

```
data(dolph.haps)
library(ape)

lowFreqSubs(as.DNAbin(dolph.haps))
```

---

maf *Minimum Allele Frequencies*

Description

Calculate minimum allele frequencies for each locus.

Usage

```
maf(g, by.strata = FALSE)
```

Arguments

g	a gtypes object.
by.strata	logical. If TRUE every element in the return list is a three dimensional array where the third dimension contains frequencies and proportions for each stratum.

Value

A vector or array of minimum allele frequencies at each locus.

Author(s)

Eric Archer <eric.archer@noaa.gov>

See Also

alleleFreqs

Examples

```
data(msats.g)

maf(msats.g)
```

mafft *MAFFT Alignment*

Description

Align a set of sequences using the MAFFT executable.

Usage

```
mafft(x, run.label = "align.mafft", delete.output = TRUE, op = 3,
  ep = 0.123, maxiterate = 0, quiet = FALSE, num.cores = 1,
  opts = "--auto")
```

Arguments

x	a list or a matrix of DNA sequences (see write.dna).
run.label	label for output alignment FASTA file.
delete.output	logical. Delete output alignment FASTA file?
op	gap opening penalty.
ep	offset value, which works like gap extension penalty.
maxiterate	number cycles of iterative refinement are performed.
quiet	logical. Run MAFFT quietly?
num.cores	number of cores to be used. Passed to MAFFT argument --thread.
opts	character string other options to provide to command line.

Value

a DNAbin object with aligned sequences.

Note

MAFFT is not included with strataG and must be downloaded separately. Additionally, it must be installed such that it can be run from the command line in the current working directory. See the vignette for external.programs for installation instructions.

Author(s)

Eric Archer <eric.archer@noaa.gov>

References

Katoh, M., Kumar, M. 2002. MAFFT: a novel method for rapid multiple sequence alignment based on fast Fourier transform. Nucleic Acids Res. 30:3059-3066.
Available at: http://mafft.cbrc.jp/alignment/software

Examples

```
## Not run:
data(dolph.seqs)
dolph.aln <- mafft(dolph.seqs, op = 3, ep = 2)
dolph.aln

## End(Not run)
```

`mavericKRun` *Run MavericK*

Description

Run MavericK clustering algorithm

Usage

```
mavericKRun(g, params = NULL, label = "MavericK_files",
  data_fname = "data.txt", param_fname = "parameters.txt")
```

Arguments

g	a gtypes object.
params	a list specifying parameters for MavericK. All parameters are available and can be specified by partial matching. The function will automatically specify parameters related to data formatting (data, headerRow_on, missingData, ploidy, ploidyCol_on, popCol_on), so those will be ignored. For a full list of available parameters and their definitions, see the MavericK documentation distributed with the program.
label	folder where input and output files will be written to.
data_fname	file name of data input file.
param_fname	file name of parameters file.

Note

MavericK is not included with `strataG` and must be downloaded separately. It can be obtained from http://www.bobverity.com/. Additionally, it must be installed such that it can be run from the command line in the current working directory. See the vignette for `external.programs` for OS-specific installation instructions.

Author(s)

Eric Archer <eric.archer@noaa.gov>

References

Robert Verity and Richard Nichols. (2016) Estimating the number of subpopulations (K) in structured populations. Genetics

Robert Verity and Richard Nichols. (2016) Documentation for MavericK software: Version 1.0

`mega` *Read and Write MEGA*

Description

Read and write MEGA formatted files.

Usage

```
read.mega(file)

write.mega(g, file = NULL, label = NULL, line.width = 60, locus = 1)
```

Arguments

`file`	a MEGA-formatted file of sequences.
`g`	a gtypes object.
`label`	label for MEGA filename (.meg). If `NULL`, the gtypes description is used if present.
`line.width`	width of sequence lines.
`locus`	number or name of locus to write.

Value

for `read.mega`, a list of:

title title of MEGA file

dna.seq DNA sequences in `DNAbin` format

Author(s)

Eric Archer `<eric.archer@noaa.gov>`

References

Sudhir Kumar, Glen Stecher, and Koichiro Tamura (2015) MEGA7: Molecular Evolutionary Genetics Analysis version 7.0. Molecular Biology and Evolution (submitted). Available at: http://www.megasoftware.net

`mostDistantSequences` *Most Distant Sequences*

Description

Finds the set of sequences that have the greatest mean pairwise distance and smallest variance of pairwise distances.

Usage

```
mostDistantSequences(x, num.seqs = NULL, model = "raw",
  pairwise.deletion = TRUE)
```

Arguments

`x`	a `DNAbin` object.
`num.seqs`	number of sequences to return.
`model`	a character string specifying the evolutionary model to be used. See dist.dna for more information.
`pairwise.deletion`	a logical indicating whether to delete sites with missing data. See dist.dna for more information.

Value

a vector of the sequence names that are have the greatest mean pairwise distance and smallest variance of pairwise distances. The names are returned in order from most to least distant.

Author(s)

Eric Archer <eric.archer@noaa.gov>

Examples

```
library(ape)
data(dolph.haps)

mostDistantSequences(as.DNAbin(dolph.haps))
```

mostRepresentativeSequences
Representative Sequences

Description

Finds the set of sequences that represent the requested number of clusters.

Usage

```
mostRepresentativeSequences(x, num.seqs = NULL, model = "raw",
  pairwise.deletion = TRUE)
```

Arguments

x	a DNAbin object.
num.seqs	number of sequence names to return.
model	a character string specifying the evolutionary model to be used. See dist.dna for more information.
pairwise.deletion	a logical indicating whether to delete sites with missing data. See dist.dna for more information.

Value

a vector of the sequence names.

Author(s)

Eric Archer <eric.archer@noaa.gov>

Examples

```
library(ape)
data(dolph.seqs)

mostRepresentativeSequences(as.DNAbin(dolph.seqs))
```

mRatio *M ratio*

Description

Calculate Garza-Williamson M ratio (bottleneck) statistic for microsattelite data.

Usage

```
mRatio(g, by.strata = TRUE, rpt.size = 8:2)
```

Arguments

g	a gtypes object.
by.strata	calculate ratio for each stratum separately?
rpt.size	set of values to check for allele repeat size. Function will use the largest common denominator found in this vector or return NA.

Value

If by.strata = TRUE, a matrix with loci as rows and strata as columns, otherwise a vector with the statistic for each locus over all samples.

Note

The function will only compute the metric for microastellite loci, which is defined as loci with allele labels that can be converted to numeric values in their entirety and have a fixed repeat size. NA is returned for all others.

Author(s)

Eric Archer <eric.archer@noaa.gov> (adapted from code by Sean Hoban)

References

Garza, J.C. and E.G. Williamson. 2001. Detection of reduction in population size using data form microsatellite loci. Molecular Ecology 10(2):305-318.

Examples

```
data(msats.g)

m.by.strata <- mRatio(msats.g, TRUE)
m.by.strata

m.overall <- mRatio(msats.g, FALSE)
m.overall
```

msats.g *Dolphin Microsatellite gtypes Object*

Description

A gtypes object of 126 samples and 4 microsatellite loci

Usage

```
data(msats.g)
```

Format

gtypes

References

Lowther-Thieleking J.L., F.I. Archer, A.R. Lang, and D.W. Weller. 2015. Genetic variation of coastal and offshore bottlenose dolphins, Tursiops truncatus, in the eastern North Pacific Ocean. Marine Mammal Science 31:1-20

neiDa *Nei's Da*

Description

Calcuate frequency-based Nei's Da for haploid or diploid data.

Usage

```
neiDa(g)
```

Arguments

g	a gtypes object.

Details

Returns Nei's Da for each pair of strata.

Author(s)

Eric Archer <eric.archer@noaa.gov>

References

Nei et al 1983 Accuracy of Estimated Phylogenetic Trees from Molecular Data. J Mol Evol 19:153-170 (eqn 7)
Nei, M., and S. Kumar (2000) Molecular Evolution and Phylogenetics. Oxford University Press, Oxford. (pp. 268, eqn 13.6)

Examples

```
data(msats.g)

neiDa(msats.g)
```

`nucleotideDivergence` *Nucleotide Divergence*

Description

Calculate Nei's dA between strata, and distributions of between- and within-strata nucleotide divergence (sequence distance).

Usage

```
nucleotideDivergence(g, probs = c(0, 0.025, 0.5, 0.975, 1), model = "raw",
  ...)
```

Arguments

`g`	a gtypes object.
`probs`	a numeric vector of probabilities of the pairwise distance distributions with values in `0:1`.
`model`	evolutionary model to be used. see `dist.dna` for options.
`...`	other arguments passed to `dist.dna`.

Value

a list with summaries of the `within` and `between` strata pairwise distances including Nei's dA.

Author(s)

Eric Archer <eric.archer@noaa.gov>

References

Nei, M., and S. Kumar (2000) Molecular Evolution and Phylogenetics. Oxford University Press, Oxford. (dA: pp. 256, eqn 12.67)

Examples

```
data(dolph.strata)
data(dolph.seqs)
strata <- dolph.strata$fine
names(strata) <- dolph.strata$ids
dloop <- sequence2gtypes(dolph.seqs, strata, seq.names = "dLoop")

nucleotideDivergence(dloop)
```

---

nucleotideDiversity *Nucleotide Diversity*

Description

Calculate nucleotide diversity for set of haplotypes.

Usage

```
nucleotideDiversity(x, bases = c("a", "c", "g", "t"), simplify = TRUE)
```

Arguments

x	a set of sequences or a gtypes object with sequences.
bases	nucleotides to consider when calculating diversity.
simplify	if TRUE and only one loci exists, return a vector, otherwise, a list of vectors with one element per locus will be returned.

Value

Nucleotide diversity by site.

Author(s)

Eric Archer <eric.archer@noaa.gov>

Examples

```
data(dolph.strata)
data(dolph.seqs)
strata <- dolph.strata$fine
names(strata) <- dolph.strata$ids
dloop <- sequence2gtypes(dolph.seqs, strata, seq.names = "dLoop")

nucleotideDiversity(dloop)
```

numAlleles *Number of Alleles*

Description

Return the number of alleles for each locus.

Usage

```
numAlleles(g)
```

Arguments

g	a gtypes object.

Value

vector of number of alleles per locus.

Note

If g is a haploid object with sequences, make sure to run `labelHaplotypes` if sequences aren't already grouped by haplotype.

Author(s)

Eric Archer <eric.archer@noaa.gov>

Examples

```
data(msats.g)

numAlleles(msats.g)
```

numericSNPmat *Create a numeric SNP matrix*

Description

Create a matrix of SNPs coded as 0, 1, 2, where 0 and 2 are the two homozygotes and 1 is the heterozygote.

Usage

```
numericSNPmat(g)
```

Arguments

g	a gtypes object.

Value

a data.frame with all biallelic loci recoded.

Note

g must be a diploid gtypes object and have at least one locus with one or two alleles.

Author(s)

Eric Archer <eric.archer@noaa.gov>

---

numGenotyped *Number of Individuals Genotyped*

Description

Return the number of individuals genotyped for each locus.

Usage

```
numGenotyped(g)
```

Arguments

g	a gtypes object.

Value

vector of number of alleles per locus.

Author(s)

Eric Archer <eric.archer@noaa.gov>

Examples

```
data(msats.g)

numGenotyped(msats.g)
```

numMissing *Number Missing Data*

Description

Calculate the number of individuals with missing data by locus.

Usage

```
numMissing(g, prop = FALSE)
```

Arguments

g	a gtypes object.
prop	logical determining whether to return proportion missing.

Value

a vector of loci with number (or, if `prop = TRUE`, the proportion) of individuals missing data for at least one allele.

Author(s)

Eric Archer <eric.archer@noaa.gov>

Examples

```
data(msats.g)

numMissing(msats.g)
numMissing(msats.g, prop = TRUE)
```

permuteStrata *Permute strata*

Description

Permute the strata slot within a gtypes object.

Usage

```
permuteStrata(g)
```

Arguments

g	a gtypes object.

Value

a gtypes object with the strata randomly permuted.

Author(s)

Eric Archer <eric.archer@noaa.gov>

Examples

```
data(msats.g)
msats.g <- stratify(msats.g, "fine")
summary(msats.g)

ran.msats <- permuteStrata(msats.g)
summary(ran.msats)
```

---

phase *PHASE*

Description

Run PHASE to estimate the phase of loci in diploid data.

Usage

```
phase(g, loci, positions = NULL, type = NULL, num.iter = 1e+05,
  thinning = 100, burnin = 1e+05, model = "new", ran.seed = NULL,
  final.run.factor = NULL, save.posterior = FALSE, in.file = "phase_in",
  out.file = "phase_out", delete.files = TRUE)

phaseReadSample(out.file, type)

phaseReadPair(out.file)

phaseWrite(g, loci, positions = NULL, type = rep("S", length(loci)),
  in.file = "phase_in")

phasePosterior(ph.res, keep.missing = TRUE)

phaseFilter(ph.res, thresh = 0.5, keep.missing = TRUE)
```

Arguments

g	a gtypes object.

loci	vector or data.frame of loci in 'g' that are to be phased. If a data.frame, it should have columns named locus (name of locus in 'g'), group (number identifying loci in same linkage group), and position (integer identifying location of each locus in a linkage group).
positions	position along chromosome of each locus.
type	type of each locus.
num.iter	number of PHASE MCMC iterations.
thinning	number of PHASE MCMC iterations to thin by.
burnin	number of PHASE MCMC iterations for burnin.
model	PHASE model type.
ran.seed	PHASE random number seed.
final.run.factor	optional.
save.posterior	logical. Save posterior sample in output list?
in.file	name to use for PHASE input file.
out.file	name to use for PHASE output files.
delete.files	logical. Delete PHASE input and output files when done?
ph.res	result from phase.run.
keep.missing	logical. T = keep missing data from original data set. F = Use estimated genotypes from PHASE.
thresh	minimum probability for a genotype to be selected (0.5 - 1).

Details

phase	runs PHASE assuming that the executable is installed properly and available on the command line.
phaseWrite	writes a PHASE formatted file.
phaseReadPair	reads the '_pair' output file.
phaseReadSample	reads the '_sample' output file.
phaseFilter	filters the result from phase.run to extract one genotype for each sample.
phasePosterior	create a data.frame all genotypes for each posterior sample.

Value

phase a list containing:

locus.name	new locus name, which is a combination of loci in group.
gtype.probs	a data.frame listing the estimated genotype for every sample along with probability.
orig.gtypes	the original gtypes object for the composite loci.
posterior	a list of num.iter data.frames representing posterior sample of genotypes for each sample.

phaseWrite a list with the input filename and the gtypes object used.

phaseReadPair a data.frame of genotype probabilities.

phaseReadSample a list of data.frames representing the posterior sample of genotypes for one set of loci for each sample.

phaseFilter a matrix of genotypes for each sample.

phasePosterior a list of data.frames representing the posterior sample of all genotypes for each sample.

Note

PHASE is not included with `strataG` and must be downloaded separately. Additionally, it must be installed such that it can be run from the command line in the current working directory. See the vignette for `external.programs` for installation instructions.

Author(s)

Eric Archer <eric.archer@noaa.gov>

References

Stephens, M., and Donnelly, P. (2003). A comparison of Bayesian methods for haplotype reconstruction from population genotype data. American Journal of Human Genetics 73:1162-1169. Available at: http://stephenslab.uchicago.edu/software.html#phase

Examples

```
## Not run:
data(bowhead.snps)
data(bowhead.snp.position)
snps <- df2gtypes(bowhead.snps, ploidy = 2, description = "Bowhead SNPS")
summary(snps)

# Run PHASE on all data
phase.results <- phase(snps, bowhead.snp.position, num.iter = 100,
  save.posterior = FALSE)

# Filter phase results
filtered.results <- phaseFilter(phase.results, thresh = 0.5)

# Convert phased genotypes to gtypes
ids <- rownames(filtered.results)
strata <- bowhead.snps$Stock[match(ids, bowhead.snps$LABID)]
filtered.df <- cbind(id = ids, strata = strata, filtered.results)
phased.snps <- df2gtypes(filtered.df, ploidy = 2, description = "Bowhead phased SNPs")
summary(phased.snps)

## End(Not run)
```

popGenEqns *Population Genetics Equations*

Description

Collection of classical population genetics equations.

Usage

```
wrightFst(Ne, dispersal, gen.time, ploidy)

numGensEq(fst, Ne, gen.time)

fstToNm(fst, ploidy)

expectedNumAlleles(n, theta, ploidy)
```

Arguments

Ne	Effective population size.
dispersal	Migration rate in terms of probability of an individual migrating in a generation.
gen.time	Number of generations since ancestral population.
ploidy	Ploidy of the locus.
fst	value of Fst at equilibrium.
n	Sample size.
theta	Product of effective population size (Ne) and mutation rate (mu).

Details

wrightFst Calculate Wright's Fst from Ne, dispersal, and generation time.

numGensEq Calculate the number of generations to equilibrium based on a an ideal Wright model.

fstToNm Calculate Nm (number of migrants per generation) for a given value of Fst.

expectedNumAlleles Calculate the expected number of alleles in a sample of a given size and value of theta.

Value

wrightFst

numGensEq

fstToNm

expectedNumAlleles a two element vector with the expected number of alleles (num.alleles) and variance (var.num.alleles).

Author(s)

Eric Archer <eric.archer@noaa.gov>

References

Ewens, W. 1972. The sampling theory of selectively neutral alleles. Theoretical Population Biology 3:87-112. Eqns. 11 and 24.

Examples

```
dispersal <- seq(0.05, 0.8, by = 0.05)
fst <- wrightFst(100, dispersal, 20, 2)
plot(dispersal, fst, type = "l")

numGensEq(0.15, 100, 20)
numGensEq(0.3, 100, 20)
numGensEq(0.15, 50, 20)

fst <- seq(0.001, 0.2, length.out = 100)
Nm <- fstToNm(fst, 2)
plot(fst, Nm, type = "l")

expectedNumAlleles(20, 1, 2)
# double the samples
expectedNumAlleles(40, 1, 2)
# for a haploid locus
expectedNumAlleles(40, 1, 1)
# double theta
expectedNumAlleles(40, 2, 1)
```

---

popStructStat *Population structure statistics*

Description

Population structure statistics

Usage

```
Hstats(g)

statChi2(g, nrep = NULL, strata.mat = NULL, keep.null = FALSE, ...)

statFis(g, nrep = NULL, strata.mat = NULL, keep.null = FALSE, ...)

statFst(g, nrep = NULL, strata.mat = NULL, keep.null = FALSE, ...)
```

```
statFstPrime(g, nrep = NULL, strata.mat = NULL, keep.null = FALSE, ...)

statGst(g, nrep = NULL, strata.mat = NULL, keep.null = FALSE, ...)

statGstPrime(g, nrep = NULL, strata.mat = NULL, keep.null = FALSE,
  prime.type = c("nei", "hedrick"), ...)

statGstDblPrime(g, nrep = NULL, strata.mat = NULL, keep.null = FALSE, ...)

statJostD(g, nrep = NULL, strata.mat = NULL, keep.null = FALSE, ...)

statPhist(g, nrep = NULL, strata.mat = NULL, keep.null = FALSE,
  model = "K80", gamma = FALSE, pairwise.deletion = TRUE, ...)
```

Arguments

g	a gtypes object.
nrep	number specifying number of permutation replicates to use for permutation test.
strata.mat	an optional matrix of permuted stratifications. See Notes for more details. Ignored if nrep is not NULL.
keep.null	logical. Keep the null distribution from the permutation test?
...	optional arguments passed to or from other functions.
prime.type	type of G'st to calculate. Can be "nei" or "hedrick".
model, gamma, pairwise.deletion	parameters passed to dist.dna. Note that defaults for these arguments (in particular model) are the same as in dist.dna.

Value

A list with three elements:

stat.name the name of the statistic.

result a vector of the statistic estimate and the p-value, if replicates were conducted.

null.dist a vector of the null distribution from the permutations.

Note

If strata.mat is provided, it must be a numeric matrix of integers from 0 to k - 1, where k is the number of strata. Each column is a separate permutation and the first column is assumed to represent the original stratification. If not provided (strata.mat = NULL), stratification is taken from g. This argument is primarily used internally by popStructTest.

Author(s)

Eric Archer <eric.archer@noaa.gov>

References

Hstats Nei, M. and R.K. Chesser. 1983. Estimation of fixation indices and gene diversities. Ann. Hum. Genet. 47:253-259.

`popStructTest` *Population Differentiation Tests*

Description

Conduct overall and/or pairwise tests of population differentiation.

Usage

```
popStructTest(g, nrep = 1000, stats = "all", type = c("both", "overall",
  "pairwise"), keep.null = FALSE, quietly = FALSE, max.cores = NULL,
  write.output = FALSE, ...)

overallTest(g, nrep = 1000, stats = "all", keep.null = FALSE,
  quietly = FALSE, max.cores = NULL, ...)

pairwiseTest(g, nrep = 1000, stats = "all", keep.null = FALSE,
  quietly = FALSE, max.cores = NULL, ...)

statList(stats = "all")
```

Arguments

`g`	a `gtypes` object.
`nrep`	number specifying number of permutation replicates to use for permutation test.
`stats`	a character vector or list of functions specifying which anlayses to conduct. If characters, then valid possible choices are: "phist", "fst", "fst.prime", "fis", "gst", "gst.prime", "gst.dbl.prime", "d", or "chi2", or "all". If a list, then functions must be a valid population structure function (see `popStructStat`) taking a gtypes object and returning a named statistic estimate.
`type`	character determining type of test to conduct. Can be "overall", "pairwise", or "both". If "pairwise" or "both" are chosen and there are only two strata, then only an overall test will be conducted.
`keep.null`	logical. Keep the null distribution from the permutation test?
`quietly`	logical. Print progress and results?
`max.cores`	The maximum number of cores to use to distribute separate statistics over. Default (NULL) sets value to what is reported by `detectCores - 1`. Any value greater than this will be set to this value. If `detectCores` reports `NA`, `max.cores` will be set to 1.
`write.output`	logical. Write a .csv file with results?
`...`	other parameters to be passed to population differentiation functions.

Value

overall a list containing:

`strata.freq` a vector of the sample sizes for each stratum

`result` a matrix with the statistic estimate and p-value for each statistic

`null.dist` a matrix with the null distributions for each statistic

pairwise a list containing:

`result` a data.frame with the result of each pairwise comparison on each row

`pair.mat` a list with a pairwise matrix for each statistic. Values in lower left are the statistic estimate, and upper right are p-values

`null.dist` a matrix with the null distributions for each statistic

Note

On multi-core systems, runs of separate statistics are automatically distributed over as many cores as available (minus one). This can be controlled by the `max.cores` argument if less core usage is desired.

Author(s)

Eric Archer <eric.archer@noaa.gov>

Examples

```
data(msats.g)
msats.g <- stratify(msats.g, "fine")

# Just an overall Chi-squared test
ovl <- overallTest(msats.g, stats = "chi2", nrep = 100)
ovl

#' Just a pairwise test for Gst
pws <- pairwiseTest(msats.g, stats = list(statGst), nrep = 100)
pws

## Not run:
#' Both overall and pairwise tests for Fst and F'st
full <- popStructTest(msats.g, stats = c("fst", "fst.prime"))
print(full$overall)
print(full$pairwise)

## End(Not run)
```

privateAlleles *Private Alleles*

Description

The number of private alleles in each strata and locus.

Usage

```
privateAlleles(g)
```

Arguments

g	a gtypes object.

Value

matrix with the number of private alleles in each strata at each locus. This is the number of alleles only present in one stratum.

Author(s)

Eric Archer <eric.archer@noaa.gov>

See Also

propUniqueAlleles

Examples

```
data(msats.g)

privateAlleles(msats.g)
```

propUniqueAlleles *Proportion Unique Alleles*

Description

Calculate the proportion of alleles that are unique.

Usage

```
propUniqueAlleles(g)
```

Arguments

g	a gtypes object.

Value

a vector of the proportion of unique (occuring only in one individual) alleles for each locus.

Note

If g is a haploid object with sequences, make sure to run `labelHaplotypes` if sequences aren't already grouped by haplotype.

Author(s)

Eric Archer <eric.archer@noaa.gov>

See Also

privateAlleles

Examples

```
data(msats.g)

propUniqueAlleles(msats.g)
```

---

qaqc *Quality Assurance/Quality Control*

Description

Conducts a suite of QA/QC tests. Summarizes missing data and homozygosity by individual and locus, and looks for duplicate genotypes (see `dupGenotypes`). For sequence data, identifies low frequency substitutions (see `lowFreqSubs`), and computes sequence likelihoods (see `sequenceLikelihoods`).

Usage

```
qaqc(g, label = NULL, ...)
```

Arguments

g	a gtypes object.
label	optional label for output folder and prefix for files.
...	optional arguments to pass on to summary functions.

Value

Files are written for by-sample and by-locus summaries, and duplicate genotypes if any are found. If sequences are present, files are written identifying low frequency substitutions and sequence likelihoods.
The return value is a list with the following elements:

by.sample data.frame of by-sample summaries

`by.locus` data.frame of by-locus summaries

`dup.df` data.frame identifying potential duplicates

`by.seq` list of low frequency substitutions and haplotype likelihoods for each gene

Author(s)

Eric Archer <eric.archer@noaa.gov>

See Also

summarizeSamples, summarizeLoci, dupGenotypes, lowFreqSubs, sequenceLikelihoods

---

readGenData *Read Genetic Data*

Description

A wrapper for read.csv that sets common values for missing data and removes blank lines.

Usage

```
readGenData(file, na.strings = c(NA, "NA", "", " ", "?", "."), ...)
```

Arguments

file	filename of .csv file.
na.strings	see read.table.
...	other arguments passed to read.table.

Value

a data.frame.

Author(s)

Eric Archer <eric.archer@noaa.gov>

removeSequences *Remove Sequences*

Description

Remove sequences not used by samples listed in `@data` slot.

Usage

```
removeSequences(g)
```

Arguments

g	a gtypes object.

Value

a new gtypes object with unused sequences removed.

Author(s)

Eric Archer <eric.archer@noaa.gov>

sequence2gtypes *Convert Sequences To* gtypes

Description

Create a gtypes object from sequence data.

Usage

```
sequence2gtypes(x, strata = NULL, seq.names = NULL, schemes = NULL,
  description = NULL, other = NULL)
```

Arguments

x	DNA sequences as a character matrix, a DNAbin object, or multidna object.
strata	a vector or factor giving stratification for each sequence. If not provided all individuals are assigned to the same stratum (Default).
seq.names	names for each set of sequences. If not provided default names are generated.
schemes	an optional data.frame of stratification schemes.
description	an optional label for the object.
other	a slot to carry other related information - unused in package analyses.

Value

a gtypes object.

Author(s)

Eric Archer <eric.archer@noaa.gov>

Examples

```
#--- create a haploid sequence (mtDNA) gtypes object
data(dolph.strata)
data(dolph.seqs)
strata <- dolph.strata$fine
names(strata) <- dolph.strata$ids
dloop.fine <- sequence2gtypes(dolph.seqs, strata, seq.names = "dLoop",
description = "dLoop: fine-scale stratification")
```

---

sequenceLikelihoods *Sequence Likelihoods*

Description

Calculate likelihood of each sequence based on gamma distribution of pairwise distances.

Usage

```
sequenceLikelihoods(x, model = "N", pairwise.deletion = FALSE, n = NULL,
  ...)
```

Arguments

x	a DNAbin object.
model	a character string specifying the evolutionary model to be used. Passed to dist.dna.
pairwise.deletion	a logical indicating whether to delete the sites with missing data in a pairwise way. Passed to dist.dna.
n	number of sequences with lowest delta(log-likelihoods) to plot. Defaults to all sequences Set to 0 to supress plotting.
...	arguments passed from other functions (ignored).

Details

Fits a Gamma distribution to the pairwise distances of sequences and calculates the log-likelihood for each (sum of all pairwise log-likelihoods for that sequence). Sequences that are extremely different from all others will have low log-likelihoods. Values returned as delta(log-likelhoods) = difference of log-likelihoods from maximum observed values.

Value

vector of delta(log-Likelihoods) for each sequence, sorted from smallest to largest, and a plot of their distributions.

Author(s)

Eric Archer <eric.archer@noaa.gov>

Examples

```
library(ape)
data(dolph.haps)

sequenceLikelihoods(as.DNAbin(dolph.haps))
```

---

show,gtypes-method *Show a gtypes object*

Description

Show a gtypes object

Usage

```
## S4 method for signature 'gtypes'
show(object)
```

Arguments

object	a gtypes object.

Author(s)

Eric Archer <eric.archer@noaa.gov>

simGammaHaps *Simulate Haplotypes*

Description

Simulate a haplotypic frequency distribution based on a specified gamma distribution.

Usage

```
simGammaHaps(pop.size, num.haps, shape, scale, return.freq = TRUE,
  plot = TRUE)
```

Arguments

pop.size	size of population.
num.haps	number of haplotypes to generate.
shape, scale	parameters of Gamma distribution (see dgamma).
return.freq	logical. Return frequency table of haplotypes? If FALSE return vector of haplotypes.
plot	logical. Show plot of haplotypic frequency distribution?

Value

Frequency table of haplotypes.

Author(s)

Eric Archer <eric.archer@noaa.gov>

Examples

```
haps <- simGammaHaps(1000, 15, 1, 2.5)
print(haps)
```

strataGUI *strataG GUI*

Description

A graphical user interface for creating gtypes objects, conducting quality control analyses, and analyses of population structure.

Usage

```
strataGUI()
```

Author(s)

Eric Archer <eric.archer@noaa.gov>

`strataSplit` *Split Strata*

Description

Return a list of gtypes for each stratum.

Usage

```
strataSplit(g, strata = NULL, remove.sequences = FALSE)
```

Arguments

g	a gtypes object.
strata	a character vector giving a subset of strata to select. If NULL then a list with all strata is created.
remove.sequences	logical. If TRUE any sequences not referenced in selected samples will not be in the returned object.

Value

A named list where each element is a gtypes object for a single stratum in g.

Author(s)

Eric Archer <eric.archer@noaa.gov>

Examples

```
data(msats.g)

# Proportion of unique alleles in each stratum
msats.list <- strataSplit(msats.g)
sapply(msats.list, propUniqueAlleles)
```

`stratify` *Stratify gtypes*

Description

Choose a new stratification scheme from the `schemes` slot in a gtypes object.

Usage

```
stratify(g, scheme = NULL, drop = TRUE)
```

Arguments

`g`	a gtypes object.
`scheme`	either the column name of a stratification scheme stored in the data.frame of the `schemes` slot of g, or a vector or factor identifying which stratum each sample belongs to.
`drop`	remove samples not assigned to a stratum? (those assigned `NA` in stratification scheme)

Value

A new gtypes object with an updated `strata` slot.

Note

If `scheme` is a vector or factor and has names, then the they will be used to match with `indNames` of `g`. Otherwise `scheme` should be the same length as the number of samples in `g` or values in `scheme` will be recycled as necessary.

Author(s)

Eric Archer `<eric.archer@noaa.gov>`

See Also

schemes

Examples

```
data(msats.g)
msats.g

broad.msats <- stratify(msats.g, "broad")
broad.msats
```

structure *STRUCTURE*

Description

Run STRUCTURE to assess group membership of samples.

Usage

```
structureRun(g, k.range = NULL, num.k.rep = 1, label = NULL,
  delete.files = TRUE, exec = "structure", ...)

structureWrite(g, label = NULL, maxpops = nlevels(strata(g)),
  burnin = 1000, numreps = 1000, noadmix = TRUE, freqscorr = FALSE,
  randomize = TRUE, seed = 0, pop.prior = NULL, locpriorinit = 1,
  maxlocprior = 20, gensback = 2, migrprior = 0.05,
  pfrompopflagonly = TRUE, popflag = NULL, ...)

structureRead(file, pops = NULL)
```

Arguments

g	a gtypes object.
k.range	vector of values to for maxpop in multiple runs. If set to NULL, a single STRUCTURE run is conducted with maxpops groups. If specified, do not also specify maxpops.
num.k.rep	number of replicates for each value in k.range.
label	label to use for input and output files
delete.files	logical. Delete all files when STRUCTURE is finished?
exec	name of executable for STRUCTURE. Defaults to "structure".
...	arguments to be passed to structure.write.
maxpops	number of groups.
burnin	number of iterations for MCMC burnin.
numreps	number of MCMC replicates.
noadmix	logical. No admixture?
freqscorr	logical. Correlated frequencies?
randomize	randomize.
seed	set random seed.
pop.prior	a character specifying which population prior model to use: "locprior" or "usepopinfo".
locpriorinit	parameterizes locprior parameter *r* - how informative the populations are. Only used when pop.prior = "locprior".

`maxlocprior` specifies range of locprior parameter *r*. Only used when `pop.prior` = "locprior".

`gensback` integer defining the number of generations back to test for immigrant ancestry. Only used when `pop.prior` = "usepopinfo".

`migrprior` numeric between 0 and 1 listing migration prior. Only used when `pop.prior` = "usepopinfo".

`pfrompopflagonly` logical. update allele frequencies from individuals specified by `popflag`. Only used when `pop.prior` = "usepopinfo".

`popflag` a vector of integers (0, 1) or logicals identifiying whether or not to use strata information. Only used when `pop.prior` = "usepopinfo".

`file` name of the output file from STRUCTURE.

`pops` vector of population labels to be used in place of numbers in STRUCTURE file.

Value

`structure.run` a list where each element is a list with results from `structure.read` and a vector of the filenames used

`structure.write` a vector of the filenames used by STRUCTURE

`structure.read` a list containing:

- `summary` new locus name, which is a combination of loci in group
- `q.mat` data.frame of assignment probabilities for each id
- `prior.anc` list of prior ancestry estimates for each individual where population priors were used
- `files` vector of input and output files used by STRUCTURE
- `label` label for the run

Note

STRUCTURE is not included with `strataG` and must be downloaded separately. Additionally, it must be installed such that it can be run from the command line in the current working directory. See the vignette for `external.programs` for installation instructions.

Author(s)

Eric Archer <eric.archer@noaa.gov>

References

Pritchard, J.K., M. Stephens, P. Donnelly. 2000. Inference of population structure using multilocus genotype data. Genetics 155:945-959.
http://web.stanford.edu/group/pritchardlab/structure.html

See Also

`structurePlot`, `evanno`, `clumpp`

Examples

```
## Not run:
data(msats.g)

# Run STRUCTURE
sr <- structureRun(msats.g, k.range = 1:4, num.k.rep = 10)

# Calculate Evanno metrics
evno <- evanno(sr)
evno

# Run CLUMPP to combine runs for K = 2
q.mat <- clumpp(sr, k = 3)
q.mat

# Plot CLUMPP results
structurePlot(q.mat)

## End(Not run)
```

`structurePlot` *Plot STRUCTURE Results*

Description

Plot Q-matrix from a call to `structure` or `clumpp`.

Usage

```
structurePlot(q.mat, pop.col = 3, prob.col = 4, sort.probs = TRUE,
  label.pops = TRUE, col = NULL, horiz = TRUE, type = NULL,
  legend.position = c("top", "left", "right", "bottom", "none"))
```

Arguments

`q.mat`	matrix or data.frame of assignment probabilities.
`pop.col`	column number identifying original population designations.
`prob.col`	column number of first assignment probabilities to first group. It is assumed that the remainder of columns (`prob.col:ncol(q.mat)`) contain all assignment probabilities.
`sort.probs`	logical. Sort individuals by probabilities within populations? If `FALSE` individuals will be plotted as in `q.mat`.
`label.pops`	logical. Label the populations on the plot?
`col`	colors to use for each group.
`horiz`	logical. Plot bars horizontally.

type	either "area" for stacked continuous area plot or "bar" for discrete stacked bar chart. The latter is prefered for small numbers of samples. If not specified, a bar chart will be used if there are <= 100 samples.
legend.position	the position of the legend ("top", "left","right", "bottom", or two-element numeric vector).

Value

invisibly, the ggplot object

Author(s)

Eric Archer <eric.archer@noaa.gov>

See Also

structure, clumpp

---

summarizeLoci *Locus Summaries*

Description

Compile standard by-locus summaries.

Usage

```
summarizeLoci(g, by.strata = FALSE, ...)
```

Arguments

g	a gtypes object.
by.strata	logical. If TRUE, return a list of summary matrices for each stratum.
...	arguments to be passed on to summary functions.

Value

A matrix with rows for each locus and columns containing summaries of:

num.genotyped The number of samples genotyped

prop.genotyped The proportion of samples genotyped

num.alleles The number of alleles in the locus

allelic.richness The allelic richness of the locus

prop.unique.alleles Proportion of alleles found in a single sample

expt.heterozygosity Expected heterozygosity

obsvd.heterozygosity Observed heterozygosity

Author(s)

Eric Archer <eric.archer@noaa.gov>

Examples

```
data(msats.g)
msats.g <- stratify(msats.g, "fine")

summarizeLoci(msats.g)
```

---

summarizeSamples *Sample Summaries*

Description

Compile standard by-sample summaries.

Usage

```
summarizeSamples(g, sort.by.strata = FALSE)
```

Arguments

g	a gtypes object.
sort.by.strata	logical. Sort data.frame by strata?

Value

A data.frame with rows for each sample and columns containing:

`id` The sample id

`strata` The stratum of the sample

`num.loci.missing.genotypes` The number of genotypes missing

`pct.loci.missing.genotypes` The proportion of genotypes missing

`pct.loci.homozygous` The proportion of loci homozygous

Author(s)

Eric Archer <eric.archer@noaa.gov>

Examples

```
data(msats.g)

summarizeSamples(msats.g)
```

summarizeSeqs *Sequence Summaries*

Description

Summaries for each sequence.

Usage

```
summarizeSeqs(x)
```

Arguments

x	a DNAbin object.

Value

a matrix listing the start and end positions of each sequence (excluding beginning and trailing N's), the length, the number of N's, and the number of indels.

Author(s)

Eric Archer <eric.archer@noaa.gov>

Examples

```
library(apex)
data(woodmouse)

summarizeSeqs(woodmouse)
```

summary,gtypes-method *Summarize gtypes Object*

Description

Generate a summary of a gtypes object.

Usage

```
## S4 method for signature 'gtypes'
summary(object, ...)

## S3 method for class 'gtypeSummary'
print(x, ...)
```

Arguments

object	a gtypes object.
...	other arguments (ignored).
x	list from summary.gtypes

Value

a list with the following elements:

num.ind number of individuals

num.loc number of loci

num.strata number of strata

unstratified number of unstratified samples

schemes names of stratification schemes

allele.freqs a list with tables of allele frequencies by strata

strata.smry a by-strata data.frame summarizing haplotypes or loci

locus.smry a data.frame summarizing each locus for non-haploid objects, NULL for haploid objects

seq.smry a summary of the sequence length and base frequencies

Author(s)

Eric Archer <eric.archer@noaa.gov>

---

tajimasD *Tajima's D*

Description

Calculate Tajima's D for a set of sequences to test for selection.

Usage

```
tajimasD(x)
```

Arguments

x	set of DNA sequences or a haploid gtypes object with sequences.

Value

A named vector with the estimate for D and the p.value that it is different from 0.

Author(s)

Eric Archer <eric.archer@noaa.gov>

References

Tajima, F. 1989. Statistical method for testing the neutral mutation hypothesis by DNA polymorphism. Genetics 123:585-595.

Examples

```
data(dolph.seqs)

tajimasD(dolph.seqs)
```

---

theta *Theta*

Description

Calculate theta from heterozygosity of each locus.

Usage

```
theta(g)
```

Arguments

g	a gtypes object.

Details

Calculates theta for each locus using the theta.h function.

Value

vector of theta values for each locus.

Author(s)

Eric Archer <eric.archer@noaa.gov>

Examples

```
data(msats.g)

theta(msats.g)
```

TiTvRatio *Transition / Transversion Ratio*

Description

Calculate transition/transversion ratio. Test substitution type of two bases.

Usage

```
TiTvRatio(x)

subType(b1, b2)

isTi(b1, b2)

isTv(b1, b2)
```

Arguments

x	a gtypes object with aligned sequences or a list of aligned DNA sequences.
b1, b2	two bases to be compared.

Value

TiTvRatio: a vector providing the number of transitions (Ti), transversions (Tv), and the transition/transversion ratio (Ti.Tv.ratio).
subType: either "ti" for transition, or "tv" for transversion.
isTi and isTv: a logical identifying whether the b1 to b2 is a transition or transversion.

Author(s)

Eric Archer <eric.archer@noaa.gov>

Examples

```
data(dolph.seqs)

TiTvRatio(dolph.seqs)

subType("a", "c")

isTi("a", "c")

isTv("a", "c")
```

trimNs *Trim N's From Sequences*

Description

Removes N's from beginning and end of sequences.

Usage

```
trimNs(x)
```

Arguments

x	a DNAbin object or list or matrix that can be coerced into one.

Value

sequences with beginning and trailing N's removed.

Author(s)

Eric Archer <eric.archer@noaa.gov>

Examples

```
 test.seqs <- list(
   A = c(rep("n", 5), "a", "c", "g", "t", rep("n", 3)),
   B = c(rep("n", 3), "a", "c", "g", "t", rep("n", 5)),
   C = c("a", "c", "g", "t", rep("n", 8))
 )

test.seqs
trimmed <- trimNs(test.seqs)
as.character(trimmed)
```

validIupacCodes *Valid IUPAC Codes*

Description

Get all possible valid IUPAC ambiguity codes for a set of nucleotide bases and ambiguity codes.

Usage

```
validIupacCodes(bases)
```

Arguments

bases	character vector of nucleotides or IUPAC codes to be checked.

Value

character vector of valid IUPAC codes for bases.

Author(s)

Eric Archer <eric.archer@noaa.gov>

Examples

```
validIupacCodes(c("c", "t", "c", "c"))

validIupacCodes(c("c", "y", "c", "c"))

validIupacCodes(c("a", "g", "t", "a"))
```

---

variableSites *Variable Sites*

Description

Identify variable sites among sequences.

Usage

```
variableSites(x, bases = c("a", "c", "g", "t", "-"))
```

Arguments

x	a gtypes object with sequences, a DNAbin object, or a list of sequences.
bases	character vector of bases to consider.

Value

A list with:

site a DNAbin object composed of variable sites.

site.freqs a matrix of base pair frequencies by site.

Author(s)

Eric Archer <eric.archer@noaa.gov>

See Also

fixedSites

Examples

```
data(dolph.haps)

variableSites(dolph.haps)
```

`write.gtypes` *Write* `gtypes`

Description

Write a gtypes object to file(s).

Usage

```
write.gtypes(g, label = NULL, folder = NULL, as.frequency = FALSE,
  by.strata = TRUE, freq.type = c("freq", "prop"), ...)
```

Arguments

`g`	a gtypes object.
`label`	label for filename(s). Default is the gtypes description if present.
`folder`	folder where file(s) should be written to. If `NULL`, files are written to current working directory.
`as.frequency`	logical indicating if haploid data should be output as frequency tables.
`by.strata`	if `as.frequency == TRUE`, calculate frequencies by strata?
`freq.type`	if `as.frequency == TRUE`, write absolute frequencies (`"freq"`) or proportions (`"prop"`).
`...`	optional arguments controlling what information is included in the genotype file and how it is formatted passed to as.matrix.

Details

Writes a comma-delimited (.csv) file of genotypes and if sequences are present, a .fasta file for each locus. If haploid and `as.frequency` is `TRUE`, then frequency tables for each locus are written to separate files.

Author(s)

Eric Archer <eric.archer@noaa.gov>

Examples

```
## Not run:
# Write microsatellites with one column per locus
data(msats.g)
write.gtypes(msats.g, one.col = TRUE)

# Write control region data as frequency tables
data(dloop.g)
write.gtypes(dloop.g, as.frequency = TRUE)

## End(Not run)
```

`write.nexus.snapp` *Write NEXUS File for SNAPP*

Description

Write NEXUS File for SNAPP

Usage

```
write.nexus.snapp(g, file = "snapp.data.nex")
```

Arguments

`g`	a gtypes object.
`file`	the filename the NEXUS file to output.

Author(s)

Eric Archer <eric.archer@noaa.gov>

Index