

Package ‘strataG’

February 28, 2020

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 2.4.905

License GNU General Public License

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

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

Depends R (>= 3.5.0)

Suggests knitr, rmarkdown, testthat, qgraph, hierfstat

Imports ape, apex (>= 1.0.3), adegenet, dplyr (>= 0.7.6), tibble (>= 1.4.2), tidyr (>= 0.8.1), purrr (>= 0.2.5), data.table (>= 1.11.4), copula, DT, genepop (>= 1.0.5), ggplot2, graphics, grid, gridExtra, Hmisc (>= 4.1), magrittr (>= 1.5), methods, pegas (>= 0.11), phangorn (>= 2.4.0), RColorBrewer, Rcpp, readr, rlang (>= 0.3.0.1), rmetasim, stats, stringi, swfscMisc (>= 1.3), utils

Collate strataG-package.R gtypes.class.R gtypes.accessors.R is.gtypes.R gtypes.initialize.R strataG-internal.R as.data.frame.gtypes.R alleleFreqs.R alleleSplit.R allelicRichness.R as.multidna.R dupGenotypes.R heterozygosity.R labelHaplotypes.R numAlleles.R numGenotyped.R numMissing.R permuteStrata.R privateAlleles.R propUniqueAlleles.R readGenData.R removeSequences.R sharedLoci.R strataSplit.R stratify.R summarizeLoci.R summarizeInds.R summarizeSeqs.R gtypes.show.R gtypes.summary.R writeGtypes.R df2gtypes.R sequence2gtypes.R baseFreqs.R createConsensus.R iupac.R expandHaplotypes.R trimNs.R fasta.R fixedSites.R variableSites.R nucleotideDiversity.R fixedDifferences.R lowFreqSubs.R sequenceLikelihoods.R nucleotideDivergence.R

popGenEqns.R freq2GenData.R gtypes2genind.R gtypes2loci.R
 gtypes2phyDat.R theta.R maf.R TiTvRatio.R simGammaHaps.R
 mostDistantSequences.R mostRepresentativeSequences.R
 write.nexus.snapp.R mega.R mRatio.R neiDa.R mafft.R arlequin.R
 ldNe.R genepop.R hweTest.R jackHWE.R LDgenepop.R fsc.input.R
 fscWrite.R fscRun.R fscRead.R sfs.R structure.R structurePlot.R
 evanno.R clumpp.R fusFs.R tajimasD.R popStructStat.R
 popStructTest.R RcppExports.R gelato.R phase.R maverickRun.R
 summarizeAll.R landscape2gtypes.R

LazyData TRUE

VignetteBuilder knitr

LinkingTo Rcpp

Encoding UTF-8

RoxygenNote 7.0.2

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 2020-02-28 07:10:02 UTC

R topics documented:

strataG-package	4
alleleFreqs	4
alleleSplit	5
allelicRichness	6
arlequin	7
as.data.frame.gtypes	9
as.multidna	11
baseFreqs	11
bowhead.snp.position	12
bowhead.snps	13
clumpp	13
createConsensus	15
df2gtypes	15
dloop.g	17
dolph.haps	18
dolph.msats	18
dolph.seqs	19
dolph.strata	19
dupGenotypes	20

evanno	21
expandHaplotypes	22
fasta	23
fixedDifferences	23
fixedSites	24
freq2GenData	25
fsc.input	26
fscRead	30
fscRun	33
fscWrite	36
fusFs	38
gelato	39
genepop	40
gtypes-class	42
gtypes.accessors	43
gtypes2genind	47
gtypes2loci	48
gtypes2phyDat	49
heterozygosity	50
hweTest	51
initialize.gtypes-method	52
is.gtypes	53
iupac	54
labelHaplotypes	55
landscape2gtypes	57
LDgenepop	57
ldNe	58
lowFreqSubs	60
maf	60
mafft	61
maverickRun	63
mega	64
mostDistantSequences	65
mostRepresentativeSequences	66
mRatio	67
msats.g	68
neiDa	68
nucleotideDivergence	69
nucleotideDiversity	70
numAlleles	71
numGenotyped	71
numMissing	72
permuteStrata	73
phase	73
popGenEqns	76
popStructStat	78
popStructTest	80
privateAlleles	82

propUniqueAlleles	83
readGenData	84
removeSequences	84
sequence2gtypes	85
sequenceLikelihoods	86
sfs	87
sharedLoci	88
simGammaHaps	89
strataSplit	90
stratify	90
structure	91
structurePlot	94
summarizeAll	96
summarizeInds	97
summarizeLoci	98
summarizeSeqs	99
summary,gtypes-method	99
tajimasD	100
theta	101
TiTvRatio	102
trimNs	103
variableSites	103
write.nexus.snapp	104
writeGtypes	105

Index **107**

strataG-package	<i>Summaries and population structure analyses of DNA sequence genotypic data</i>
-----------------	---

Description

strataG

alleleFreqs	<i>Allele Frequencies</i>
-------------	---------------------------

Description

Calculate allele frequencies or proportions for each locus.

Usage

```
alleleFreqs(g, by.strata = FALSE, type = c("freq", "prop"))
```

Arguments

g a [gtypes](#) object.
 by.strata logical determining if results should be returned by strata?
 type return counts ("freq") or proportions ("prop")

Value

A list of allele frequencies for each locus. Each element is a vector (by.strata = FALSE) or matrix (by.strata = TRUE) with the frequency or proportion of each allele.

Note

If g is a haploid object with sequences, the function will run [labelHaplotypes](#) if sequences aren't already grouped by haplotype. The gtypes object used with haplotype assignments and unassigned individuals will be stored in attr(*, "gtypes").

Author(s)

Eric Archer <eric.archer@noaa.gov>

Examples

```
data(msats.g)

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

f.pop <- alleleFreqs(msats.g, TRUE, "prop")
f.pop$EV94[, "Coastal"] # Proportions of EV94 alleles 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 microsatellite 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, by.strata = FALSE)
```

Arguments

g a [gtypes](#) object.
by.strata logical - return results grouped by strata?

Value

a data frame with 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	<i>Read and Write Arlequin Files</i>
----------	--------------------------------------

Description

Read an Arlequin-formatted project input file (.arp). Convert .arp data into gtypes object. Write an input file from a gtypes object.

Usage

```
arlequinRead(file)

arp2gtypes(arp, avoid.dups = FALSE)

arlequinWrite(g, file = NULL, locus = 1)

read.arlequin(file)

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

Arguments

file	filename of an arlequin project (.arp) file. See Notes for details on how .arp files are parsed.
arp	a list of arlequin profile information and data as returned from arlequinRead.
avoid.dups	logical. Should sample identifiers be combined with strata names to avoid duplicate identifiers between strata? If set to FALSE, ids will be left unchanged, but an error will be thrown when the gtypes object is created if duplicated ids are found.
g	a gtypes object.
locus	numeric or character designation of which locus to write for haploid data.

Details

arlequinRead	parses a .arp file.
arp2gtypes	converts list from parsed .arp file to gtypes.
arlequinWrite	writes gtypes to .arp file.

Value

arlequinRead a list containing:

file	name and full path of .arp file that was read.
profile.info	list containing parameters in [[Profile]] section of .arp file. All parameters are provided. Parameters unused are set to NULL.
data.info	list containing data from [[Data]] section of .arp file. Can contain elements for haplotype.definition (a list of haplotypes) and individual.definition (a list of individuals).

arp2gtypes a `gtypes` object.

arlequinWrite the filename of the .arp file that was written.

Note

`arp2gtypes()` will not create a `gtypes` object for Arlequin projects with relative frequency data (`DataType=FREQUENCY` and `FREQUENCY=REL`). If `DataType=DNA` and `GenotypicData=0`, sequences for each haplotype or individual are assumed to be from a single locus.

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/>

Examples

```
# write test microsat data .arp file
f <- arlequinWrite(msats.g, tempfile())

# read .arp file and show structure
msats.arp <- arlequinRead(f)
print(str(msats.arp))

# convert parsed data to gtypes object
msats.arp.g <- arp2gtypes(msats.arp)
msats.arp.g

# compare to original
```



```
msats.g
```

```
as.data.frame.gtypes
```

Convert gtypes to data.frame or matrix

Description

Create a formatted data.frame or matrix 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,  
  coded.snps = FALSE,  
  ref.allele = NULL,  
  ...  
)
```

```
## 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 stratification (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)

<code>coded.snps</code>	return diploid SNPs coded as 0 (reference allele homozygote), 1 (heterozygote), or 2 (alternate allele homozygote). If this is 'TRUE', the data is diploid, and all loci are biallelic, a data frame of coded genotypes will be returned with one column per locus.
<code>ref.allele</code>	an optional vector of reference alleles for each SNP. Only used if 'coded.snps = TRUE'. If provided, it must be at least as long as there are biallelic SNPs in <code>g</code> . If named, the names must match those of all loci in <code>g</code> . If set to 'NULL' (default) the major allele at each SNP is used as the reference.
<code>...</code>	additional arguments to be passed to or from methods.

Value

A data.frame or matrix with one row per individual.

Author(s)

Eric Archer <eric.archer@noaa.gov>

See Also

[df2gtypes as.matrix](#)

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 a matrix instead
genotypes.mat <- as.matrix(msats.g)
str(genotypes.mat)
```

as.multidna	<i>Convert to multidna</i>
-------------	----------------------------

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

[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	<i>Base Frequencies</i>
-----------	-------------------------

Description

Calculate nucleotide base frequencies along a sequence.

Usage

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

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 site frequencies.
simplify	if there is a single locus, return result in a simplified form? If FALSE a list will be returned with one element per locus.

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	<i>Consensus Sequence</i>
-----------------	---------------------------

Description

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

Usage

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

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.
simplify	if there is a single locus, return result in a simplified form? If FALSE a list will be returned with one element per locus.

Value

A character vector of the consensus sequence.

Author(s)

Eric Archer <eric.archer@noaa.gov>

Examples

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

df2gtypes	<i>Convert a data.frame to gtypes</i>
-----------	---------------------------------------

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

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

[gtypes.initialize](#), [sequence2gtypes](#), [as.data.frame.gtypes](#), [gtypes2genind](#), [gtypes2loci](#), [gtypes2phyDat](#)

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)
```

Arguments

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

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.
mismatch.loci	loci where the two samples do not match.
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.

Author(s)

Eric Archer <eric.archer@noaa.gov>

Examples

```
data(msats.g)

# identify potential duplicates in Coastal strata
coastal <- msats.g[, , "Coastal"]
coastal.5 <- coastal[getIndNames(coastal)[1:5], , ]
dupes <- dupGenotypes(coastal.5)
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>

See Also

[labelHaplotypes](#)

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	<i>Read and Write FASTA</i>
-------	-----------------------------

Description

Read and write FASTA formatted files of sequences.

Usage

```
read.fasta(file)
```

```
write.fasta(x, file = NULL)
```

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 invisibly, name(s) of file(s) written

Author(s)

Eric Archer <eric.archer@noaa.gov>

fixedDifferences	<i>Fixed Differences</i>
------------------	--------------------------

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 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", "-"), simplify = TRUE)
```

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.
simplify if there is a single locus, return result in a simplified form? If FALSE a list will be returned with one element per locus.

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)
```

fsc.input

Input functions for fastsimcoal parameters

Description

These functions specify and format simulation parameters used to write fastsimcoal2 parameter or template files, parameter estimation files, parameter definition files, and site frequency spectrum files.

Usage

```
fscDeme(deme.size, sample.size, sample.time = 0, inbreeding = 0, growth = 0)  
  
fscSettingsDemes(..., ploidy = 2)  
  
fscEvent(  
  event.time = 0,  
  source = 0,  
  sink = 0,  
  prop.migrants = 1,  
  new.size = 1,  
  new.growth = 0,  
  migr.mat = 0  
)  
  
fscSettingsEvents(...)
```

```
fscSettingsMigration(...)  
  
fscBlock_dna(  
  sequence.length,  
  mut.rate,  
  recomb.rate = 0,  
  transition.rate = 1/3,  
  chromosome = 1  
)  
  
fscBlock_microsat(  
  num.loci,  
  mut.rate,  
  recomb.rate = 0,  
  gsm.param = 0,  
  range.constraint = 0,  
  chromosome = 1  
)  
  
fscBlock_snp(sequence.length, mut.rate, recomb.rate = 0, chromosome = 1)  
  
fscBlock_standard(num.loci, mut.rate, recomb.rate = 0, chromosome = 1)  
  
fscBlock_freq(mut.rate, outexp = TRUE)  
  
fscSettingsGenetics(..., num.chrom = NULL)  
  
fscEstParam(  
  name,  
  is.int = TRUE,  
  distr = c("unif", "logunif"),  
  min = NA,  
  max = NA,  
  value = NA,  
  output = TRUE,  
  bounded = FALSE,  
  reference = FALSE  
)  
  
fscSettingsEst(..., obs.sfs, rules = NULL, sfs.type = c("maf", "daf"))  
  
fscSettingsDef(mat)
```

Arguments

deme.size the number of individuals in the deme.
sample.size the number of samples to take.

sample.time	the number of generations in the past at which samples are taken.
inbreeding	the inbreeding coefficient for the deme [0:1].
growth	the growth rate of the deme.
...	a set of comma-separated values for settings. See Notes for more information.
ploidy	the desired ploidy of the final data. deme.size and sample.size will be multiplied by this value in the parameter or template file as fastsimcoal2 generates haploid data.
event.time	the number of generations before present at which the historical event happened.
source	the source deme (the first listed deme has index 0).
sink	the sink deme.
prop.migrants	the expected proportion of migrants to move from the source to the sink deme.
new.size	the new size for the sink deme, relative to its size in the previous (later in time) generation.
new.growth	the new growth rate for the sink deme.
migr.mat	the number of the new migration matrix to be used further back in time. The matrices are those supplied to the fscSettingsMigration function. The first matrix has index 0.
sequence.length	number of base pairs to use for each block.
mut.rate	per base pair or locus mutation rate.
recomb.rate	recombination rate between adjacent markers. No effect for SNPs.
transition.rate	dna: fraction of substitutions that are transitions.
chromosome	number or character identifying which chromosome the marker is on.
num.loci	number of loci to simulate.
gsm.param	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.
outexp	logical describing if the expected site frequency spectrum given the estimated parameters should be output?
num.chrom	the number of chromosomes to be simulated. If this is specified and not the same as the number of linkage blocks specified by the fscBlock_ functions, then this many chromosomes with duplicated structures will be simulated. If num.chrom = NULL, then the chromosome specification for each block will be used.
name	name of the parameter being specified. Must match a name used in one of the simulation settings functions.
is.int	logical specifying whether or not the parameter is an integer.

distr	a character string giving the distribution to use to select initial values for parameter estimation. Can be "unif" or "logunif".
min, max	minimum and maximum values for the distribution specified in <code>distr</code> .
value	character string giving the value that the complex parameter is to take.
output	logical indicating if estimates for the parameter should be output.
bounded	logical indicating whether to treat the parameter as a bounded estimate.
reference	logical indicating whether the parameter is to be used as a reference.
obs.sfs	vector, matrix, or list containing observed SFS to use for parameter estimation.
rules	character vector giving rules for the parameter estimation.
sfs.type	type of SFS to write. Can be maf or daf.
mat	numeric matrix or data frame with values of parameters to use in place of parameter names in simulation.

Note

All settings must be passed to `fscWrite` using one of the `fscSettingsXXX` functions. Most of these functions in turn take as their input comma-separated values which are the result of specific `fscXXX` functions:

fscSettingsDemes() comma-separated instances of `fscDeme()`. If names are given for each deme, these names will be used in the parsed output.

fscSettingsEvents() comma-separated instances of `fscEvent()`.

fscSettingsMigration() comma-separated migration matrices.

fscSettingsGenetics() comma-separated instances of `fscBlock_dna()`, `fscBlock_microsat()`, `fscBlock_snp()`, `fscBlock_standard()`, or `fscBlock_freq()`. SNPs are simulated as a DNA sequence with a transition rate of 1. `'fscBlock_freq()'` can only be used by itself and in parameter estimation simulations.

fscSettingsEst() comma-separated instances of `fscEstParam()` as well as site frequency spectra (`obs.sfs`) and parameter rules `rules`.

`fastsimcoal2` 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. The function `fscTutorial()` will open a detailed tutorial on the interface in your web browser.

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.

Excoffier, L., Dupanloup, I., Huerta-Sánchez, E., Sousa, V.C., and M. Foll (2013) Robust demographic inference from genomic and SNP data. *PLOS Genetics*, 9(10):e1003905.

<http://cmpg.unibe.ch/software/fastsimcoal2/>

See Also

[fscWrite](#), [fscRun](#), [fscRead](#)

Examples

```
# three demes with optional names
demes <- fscSettingsDemes(
  Large = fscDeme(10000, 10),
  Small = fscDeme(2500, 10),
  Medium = fscDeme(5000, 3, 1500)
)

# four historic events
events <- fscSettingsEvents(
  fscEvent(event.time = 2000, source = 1, sink = 2, prop.migrants = 0.05),
  fscEvent(2980, 1, 1, 0, 0.04),
  fscEvent(3000, 1, 0),
  fscEvent(15000, 0, 2, new.size = 3)
)

# four genetic blocks of different types on three chromosomes.
genetics <- fscSettingsGenetics(
  fscBlock_snp(10, 1e-6, chromosome = 1),
  fscBlock_dna(10, 1e-5, chromosome = 1),
  fscBlock_microsat(3, 1e-4, chromosome = 2),
  fscBlock_standard(5, 1e-3, chromosome = 3)
)

#' same four genetic blocks of different types with same structure repeated three times.
genetics <- fscSettingsGenetics(
  fscBlock_snp(10, 1e-6),
  fscBlock_dna(10, 1e-5),
  fscBlock_microsat(3, 1e-4),
  fscBlock_standard(5, 1e-3),
  num.chrom = 3
)
```

fscRead

Read fastsimcoal output

Description

Read arlequin formatted output or parameter estimation files generated by fastsimcoal

Usage

```
fscReadArp(
  p,
```

```

sim = c(1, 1),
marker = c("all", "snp", "microsat", "dna", "standard"),
chrom = NULL,
sep.chrom = FALSE,
drop.mono = FALSE,
as.genotypes = TRUE,
one.col = FALSE,
sep = "/",
coded.snps = FALSE
)

fscReadParamEst(p)

fscReadSFS(p, sim = 1)

```

Arguments

<code>p</code>	list of fastsimcoal input parameters.
<code>sim</code>	one or two-element numeric vector giving the number of the simulation replicate (and sub-replicate) to read. For example, <code>sim = c(3, 5)</code> will attempt to read " <code><label>_3_5.arp</code> ".
<code>marker</code>	type of marker to return.
<code>chrom</code>	numerical vector giving chromosomes to return. If <code>NULL</code> all chromosomes are returned.
<code>sep.chrom</code>	return a list of separate chromosomes?
<code>drop.mono</code>	return only polymorphic loci?
<code>as.genotypes</code>	return data as genotypes? If <code>FALSE</code> , original haploid data is returned. If <code>TRUE</code> , individuals are created by combining sequential haplotypes based on the ploidy used to run the simulation.
<code>one.col</code>	return genotypes with one column per locus? If <code>FALSE</code> , alleles are split into separate columns and designated as ".1", ".2", etc. for each locus.
<code>sep</code>	character to use to separate alleles if <code>one.col = TRUE</code> .
<code>coded.snps</code>	return diploid SNPs coded as 0 (major allele homozygote), 1 (heterozygote), or 2 (minor allele homozygote). If this is <code>TRUE</code> and <code>marker = "snp"</code> (or only SNPs are present) and the data is diploid, genotypes will be returned with one column per locus.

Value

fscReadArp Reads and parses Arlequin-formatted `.arp` output files created by `fastsimcoal2`. Returns a data frame of genotypes, with individuals created by combining haplotypes based on the stored value of ploidy specified when the simulation was run.

fscReadParamEst Reads and parses files output from a `fastsimcoal2` run conducted for parameter estimation. Returns a list of data frames and vectors containing the data from each file.

fscReadSFS Reads site frequency spectra generated from `fastsimcoal2`. Returns a list of the marginal and joint SFS, the polymorphic sites, and the estimated maximum likelihood of the SFS."

Note

fastsimcoal2 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. The function `fscTutorial()` will open a detailed tutorial on the interface in your web browser.

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.

Excoffier, L., Dupanloup, I., Huerta-Sánchez, E., Sousa, V.C., and M. Foll (2013) Robust demographic inference from genomic and SNP data. *PLOS Genetics*, 9(10):e1003905.

<http://cmpg.unibe.ch/software/fastsimcoal2/>

See Also

`fsc.input`, `fscWrite`, `fscRun`

Examples

```
## Not run:
#' # three demes with optional names
demes <- fscSettingsDemes(
  Large = fscDeme(10000, 10),
  Small = fscDeme(2500, 10),
  Medium = fscDeme(5000, 3, 1500)
)

# four historic events
events <- fscSettingsEvents(
  fscEvent(event.time = 2000, source = 1, sink = 2, prop.migrants = 0.05),
  fscEvent(2980, 1, 1, 0, 0.04),
  fscEvent(3000, 1, 0),
  fscEvent(15000, 0, 2, new.size = 3)
)

# four genetic blocks of different types on three chromosomes.
genetics <- fscSettingsGenetics(
  fscBlock_snp(10, 1e-6, chromosome = 1),
  fscBlock_dna(10, 1e-5, chromosome = 1),
  fscBlock_microsat(3, 1e-4, chromosome = 2),
  fscBlock_standard(5, 1e-3, chromosome = 3)
)

params <- fscWrite(demes = demes, events = events, genetics = genetics)

# runs 100 replicates, converting all DNA sequences to 0/1 SNPs
# will also output the MAF site frequency spectra (SFS) for all SNP loci.
```



```
params <- fscRun(params, num.sim = 100, dna.to.snp = TRUE, num.cores = 3)

# extracting only microsatellite loci from simulation replicate 1
msats <- fscReadArp(params, marker = "microsat")

# read SNPs from simulation replicate 5 with genotypes coded as 0/1
snp.5 <- fscReadArp(params, sim = 1, marker = "snp", coded.snps = TRUE)

# read SFS for simulation 20
sfs.20 <- fscReadSFS(params, sim = 20)

## End(Not run)
```

fscRun

Run fastsimcoal

Description

Run a fastsimcoal simulation.

Usage

```
fscRun(
  p,
  num.sims = 1,
  dna.to.snp = FALSE,
  max.snps = 0,
  sfs.type = c("maf", "daf"),
  nonpar.boot = NULL,
  all.sites = TRUE,
  inf.sites = FALSE,
  no.arl.output = FALSE,
  num.loops = 20,
  min.num.loops = 20,
  brentol = 0.01,
  trees = FALSE,
  num.cores = 1,
  seed = NULL,
  quiet = TRUE,
  exec = "fsc26"
)

fscCleanup(label, folder = ".")

fscTutorial()
```

Arguments

p	list of fastsimcoal input parameters and output produced by fscWrite .
num.sims	number of simulation replicates to run.
dna.to.snp	convert DNA sequences to numerical SNPs?
max.snps	maximum number of SNPs to retain.
sfs.type	type of site frequency spectrum to compute for each population sample: 'daf' = derived allele frequency (unfolded), 'maf' = minor allele frequency (folded).
nonpar.boot	number of bootstraps to perform on polymorphic sites to extract SFS.
all.sites	retain all sites? If FALSE, only polymorphic DNA sites will be returned. This includes SNP blocks as they are simulated as DNA sequences.
inf.sites	use infinite sites model? If TRUE, all mutations are retained in the output, thus the number of sites for SNPs or DNA sequences will potentially be greater than what was requested.
no.arl.output	do not output arlequin files.
num.loops	number of loops (ECM cycles) to be performed when estimating parameters from SFS. Default is 20.
min.num.loops	number of loops (ECM cycles) for which the likelihood is computed on both monomorphic and polymorphic sites. Default is 20.
brentol	Tolerance level for Brent optimization. Smaller value imply more precise estimations, but require more computation time. Default = 0.01. Value is restricted between 1e-5 and 1e-1.
trees	output NEXUS formatted coalescent trees for all replicates?
num.cores	number of cores to use. If set to NULL, the value will be what is reported by detectCores -1.
seed	random number seed for simulation.
quiet	logical indicating if fastsimcoal2 should be run in quiet mode.
exec	name of fastsimcoal executable.
label	character string of file run labels prefixes.
folder	character string giving the root working folder where input files and output resides

Value

fscRun Runs the fastsimcoal2 simulation and returns a list containing run parameters and a data frame used by [fscRead](#) to parse the genotypes generated (if Arlequin-formatted output was requested).

fscCleanup Deletes all files associated with the simulation identified by label.

Note

fastsimcoal2 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. The function `fscTutorial()` will open a detailed tutorial on the interface in your web browser.

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.

Excoffier, L., Dupanloup, I., Huerta-Sánchez, E., Sousa, V.C., and M. Foll (2013) Robust demographic inference from genomic and SNP data. *PLOS Genetics*, 9(10):e1003905.

<http://cmpg.unibe.ch/software/fastsimcoal2/>

See Also

[fsc.input](#), [fscWrite](#), [fscRead](#)

Examples

```
## Not run:
#' # three demes with optional names
demes <- fscSettingsDemes(
  Large = fscDeme(10000, 10),
  Small = fscDeme(2500, 10),
  Medium = fscDeme(5000, 3, 1500)
)

# four historic events
events <- fscSettingsEvents(
  fscEvent(event.time = 2000, source = 1, sink = 2, prop.migrants = 0.05),
  fscEvent(2980, 1, 1, 0, 0.04),
  fscEvent(3000, 1, 0),
  fscEvent(15000, 0, 2, new.size = 3)
)

# four genetic blocks of different types on three chromosomes.
genetics <- fscSettingsGenetics(
  fscBlock_snp(10, 1e-6, chromosome = 1),
  fscBlock_dna(10, 1e-5, chromosome = 1),
  fscBlock_microsat(3, 1e-4, chromosome = 2),
  fscBlock_standard(5, 1e-3, chromosome = 3)
)

params <- fscWrite(demes = demes, events = events, genetics = genetics)

# runs 100 replicates, converting all DNA sequences to 0/1 SNPs
# will also output the MAF site frequency spectra (SFS) for all SNP loci.
params <- fscRun(params, num.sim = 100, dna.to.snp = TRUE, num.cores = 3)

## End(Not run)
```

fscWrite *Write fastsimcoal2 input files*

Description

Write files necessary to run a fastsimcoal2 simulation.

Usage

```
fscWrite(
  demes,
  genetics,
  migration = NULL,
  events = NULL,
  est = NULL,
  def = NULL,
  label = "strataG.fsc",
  use.wd = FALSE
)
```

Arguments

demes	matrix of deme sampling information created by the fscSettingsDemes function.
genetics	data.frame specifying loci to simulate created by the fscSettingsGenetics function.
migration	a list of matrices giving the migration rates between pairs of demes created by the fscSettingsMigration function.
events	matrix of historical events created by the fscSettingsEvents function.
est	list of parameter estimation definitions and rules generated by the fscSettingsEst function.
def	matrix of parameter values to substitute into the model generated by the fscSettingsDef function.
label	character string used to label output files for the simulation.
use.wd	use current working directory for input and output? If FALSE then a temporary directory in the session temporary directory. Note that this directory is deleted when the R session closed. See tempdir for more information.

Value

Writes input files for fastsimcoal2 and returns a list of input parameters, input file, and input filenames. This list is the primary input to [fscRun](#).

Note

fastsimcoal2 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. The function fscTutorial() will open a detailed tutorial on the interface in your web browser.

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.

Excoffier, L., Dupanloup, I., Huerta-Sánchez, E., Sousa, V.C., and M. Foll (2013) Robust demographic inference from genomic and SNP data. *PLOS Genetics*, 9(10):e1003905.

<http://cmpg.unibe.ch/software/fastsimcoal2/>

See Also

[fsc.input](#), [fscRun](#), [fscRead](#)

Examples

```
## Not run:
#' # three demes with optional names
demes <- fscSettingsDemes(
  Large = fscDeme(10000, 10),
  Small = fscDeme(2500, 10),
  Medium = fscDeme(5000, 3, 1500)
)

# four historic events
events <- fscSettingsEvents(
  fscEvent(event.time = 2000, source = 1, sink = 2, prop.migrants = 0.05),
  fscEvent(2980, 1, 1, 0, 0.04),
  fscEvent(3000, 1, 0),
  fscEvent(15000, 0, 2, new.size = 3)
)

# four genetic blocks of different types on three chromosomes.
genetics <- fscSettingsGenetics(
  fscBlock_snp(10, 1e-6, chromosome = 1),
  fscBlock_dna(10, 1e-5, chromosome = 1),
  fscBlock_microsat(3, 1e-4, chromosome = 2),
  fscBlock_standard(5, 1e-3, chromosome = 3)
)

params <- fscWrite(demes = demes, events = events, genetics = genetics)

## End(Not run)
```

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. Statistical significance (p-values) of Fs must be calculated with case-specific simulations.

Author(s)

Eric Archer <eric.archer@noaa.gov>

References

Fu, Y-X. 1997. Statistical tests of neutrality of mutations against population growth, hitchhiking 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, num.cores = 1)
gelatoPlot(gelato.result, unknown = NULL, main = NULL)
```

Arguments

<code>g</code>	a gtypes object.
<code>unknown.strata</code>	a character vector listing to assign. Strata must occur in <code>g</code> .
<code>nrep</code>	number of permutation replicates for Fst distribution.
<code>min.sample.size</code>	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.
<code>num.cores</code>	The number of cores to use to distribute replicates over. If set to NULL, the value will be what is reported by detectCores -1.
<code>gelato.result</code>	the result of a call to <code>gelato</code> .
<code>unknown</code>	the names of the unknown strata in the <code>x\$likelihoods</code> element to create plots. If NULL one plot for each stratum is created.
<code>main</code>	main label for top of plot.#'

Value

A list with the following elements:

<code>assign.prob</code>	a data.frame of assignment probabilities.
<code>likelihoods</code>	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* 77(1):47-58

Examples

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

# Run GELATo analysis
gelato.fine <- gelato(msats.g, unk = "Offshore.South", nrep = 20, num.cores = 2)
gelato.fine

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

## End(Not run)
```

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 = NULL)
```

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 list with the filename written and a vector mapping locus names in the file to the original locus names.

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](#), [gtypes.accessors](#), [gtypes.initialize](#)

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
labelHaplotypes(dloop.g, "Hap.")

```

gtypes.accessors	gtypes <i>Accessors</i>
------------------	-------------------------

Description

Accessors for slots in [gtypes](#) objects.

Usage

```

## S4 method for signature 'gtypes'
getNumInd(x, by.strata = FALSE, ...)

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

getNumStrata(x, ...)

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

getIndNames(x, ...)

## S4 method for signature 'gtypes'
getIndNames(x, by.strata = FALSE, ...)

getLociNames(x, ...)

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

getAlleleNames(x, ...)

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

getStrataNames(x, ...)

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

getPloidy(x, ...)

```

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

getStrata(x, ...)

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

setStrata(x) <- value

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

getSchemes(x, ...)

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

setSchemes(x) <- value

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

getSequences(x, ...)

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

getDescription(x, ...)

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

setDescription(x) <- value

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

getOther(x, ...)

## S4 method for signature 'gtypes'
getOther(x, value = NULL, ...)

setOther(x, name) <- value

## S4 replacement method for signature 'gtypes'
setOther(x, name) <- value
```

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

Arguments

x	a gtypes object.
by.strata	logical - return results by strata?
...	other arguments passed from generics (ignored).
value	value being assigned by accessor.
as.haplotypes	return sequences as haplotypes? If TRUE, contents of @sequences slot are returned. If FALSE, one sequence per individual is returned.
seqName	the name (or number) of a set of sequences from the @sequences slot to return.
as.multidna	return sequences as a multidna object? If FALSE, sequences are returned as a list.
name	name of the value going into the other list.
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 is the order returned by `getIndNames`, `getLocNames`, and `getStrataNames` 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")

getNumStrata(msats.g)
getStrataNames(msats.g)
getNumLoci(msats.g)
getLociNames(msats.g)

# reassign all samples to two randomly chosen strata
new.strata <- sample(c("A", "B"), getNumInd(msats.g), rep = TRUE)
names(new.strata) <- getIndNames(msats.g)
setStrata(msats.g) <- new.strata
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)
new.strata <- sample(c("A", "B"), getNumInd(wood.g), rep = TRUE)
names(new.strata) <- getIndNames(wood.g)
setStrata(wood.g) <- new.strata
wood.g

# get the multidna sequence object
multi.seqs <- getSequences(wood.g, as.multidna = TRUE)
class(multi.seqs) # "multidna"

# get a list of DNABin objects
dnabin.list <- getSequences(wood.g)
class(dnabin.list) # "list"

# get a DNABin object of the first locus
dnabin.1 <- getSequences(wood.g)[[1]]
class(dnabin.1) # "DNABin"

# getting and setting values in the `other` slot:
getOther(dloop.g)

setOther(dloop.g, "timestamp") <- timestamp()
setOther(dloop.g, "Author") <- "Hoban Washburne"

getOther(dloop.g)
getOther(dloop.g, "timestamp")

```

```
setOther(dloop.g, "Author") <- NULL
getOther(dloop.g)
```

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	<i>Convert Between gtypes And loci objects.</i>
-------------	---

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](#), [as.matrix.gtypes](#), [gtypes2genind](#), [gtypes2loci](#)

Examples

```
data(dloop.g)

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

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

heterozygosity	<i>Heterozygosity</i>
----------------	-----------------------

Description

Calculate observed and heterozygosity.

Usage

```
heterozygosity(g, by.strata = FALSE, type = c("expected", "observed"))
```

Arguments

<code>g</code>	a gtypes object.
<code>by.strata</code>	logical - return results by strata?
<code>type</code>	return expected or observed heterozygosity

Note

If `g` is a haploid object with sequences, the value for expected heterozygosity (= haplotypic diversity) will be returned.

Author(s)

Eric Archer <eric.archer@noaa.gov>

Examples

```
data(msats.g)

# Expected heterozygosity
heterozygosity(msats.g, type = "expected")

# Observed heterozygosity by strata
heterozygosity(msats.g, FALSE, "observed")
```

hweTest	<i>Hardy-Weinberg Equilibrium</i>
---------	-----------------------------------

Description

Calculate Hardy-Weinberg equilibrium p-values.

Usage

```
hweTest(  
  g,  
  use.genepop = FALSE,  
  which = c("Proba", "excess", "deficit"),  
  enumeration = FALSE,  
  dememorization = 10000,  
  batches = 20,  
  num.rep = 5000,  
  delete.files = TRUE,  
  label = NULL  
)
```

Arguments

<code>g</code>	a gtypes object.
<code>use.genepop</code>	logical. Use GENEPOP to calculate HWE p-values? If FALSE then hw.test is used.
<code>which, enumeration, dememorization, batches</code>	parameters for GENEPOP MCMC HWE procedure as defined in test_HW .
<code>num.rep</code>	the number of replicates for the Monte Carlo procedure for hw.test or number of iterations for test_HW .
<code>delete.files</code>	logical. Delete GENEPOP files when done?
<code>label</code>	character string to use to label GENEPOP files.

Value

a vector of p-values for each locus.

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

<code>.Object</code>	the object skeleton, automatically generated when calling <code>new</code> .
<code>gen.data</code>	a vector, matrix, or <code>data.frame</code> containing the alleles at each locus. See below for more details.
<code>ploidy</code>	ploidy of the loci.
<code>ind.names</code>	an optional vector of individual sample names.
<code>sequences</code>	an optional multidna object containing sequences represented by each locus.
<code>strata</code>	an optional stratification scheme from <code>schemes</code> .
<code>schemes</code>	an optional <code>data.frame</code> of stratification schemes.
<code>description</code>	an optional description for the object.
<code>other</code>	other optional information to include.
<code>remove.sequences</code>	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`. If `schemes` has a column named 'id', it will be used to match to sample names in `gen.data`. Otherwise, if rownames are present in `schemes`, a column named 'id' will be created from them. If sequences are provided in `sequences`, then they should be named and match values in the haplotype column in `gen.data`. If multiple genes are given as a `multidna` object, it is assumed that they are in the same order as the columns in `gen.data`.

Author(s)

Eric Archer <eric.archer@noaa.gov>

See Also

[df2gtypes](#), [sequence2gtypes](#)

is.gtypes	<i>Test if object is gtypes</i>
-----------	---------------------------------

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
```

iupac

IUPAC Codes

Description

Calculate the correct IUPAC code for a vector of nucleotides.

Usage

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

validIupacCodes(bases)

iupacMat()
```

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

iupacCode	a character representing the correct IUPAC code bases.
validIupacCodes	a character vector of all valid IUPAC codes for bases.
iupacMat	a logical matrix identifying valid IUPAC codes.

Author(s)

Eric Archer <eric.archer@noaa.gov>

See Also

[validIupacCodes](#)

Examples

```

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

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

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

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

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

```

labelHaplotypes	<i>Find and label haplotypes</i>
-----------------	----------------------------------

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 <code>NULL</code> , haplotypes will be labeled with the first label from original sequences.
use.indels	logical. Use indels when comparing sequences?
...	arguments to be passed to <code>labelHaplotypes.default</code> .

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 (multidna) 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 new gtypes object with unassigned individuals stored in the @other slot in an element named 'haps.unassigned' (see [getOther](#)).

Author(s)

Eric Archer <eric.archer@noaa.gov>

See Also

[expandHaplotypes](#)

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
```

landscape2gtypes	<i>Convert Rmetasim landscape</i>
------------------	-----------------------------------

Description

‘landscape2gtypes‘ creates a gtypes object from an Rmetasim landscape object. ‘landscape2df‘ creates a data.frame.

Usage

```
landscape2gtypes(Rland)  
  
landscape2df(Rland)
```

Arguments

Rland rmetasim landscape object

Author(s)

Eric Archer <eric.archer@noaa.gov>

LDgenepop	<i>Linkage Disequilibrium</i>
-----------	-------------------------------

Description

Calculate linkage disequilibrium p-values using GENEPOP.

Usage

```
LDgenepop(  
  g,  
  dememorization = 10000,  
  batches = 100,  
  iterations = 5000,  
  delete.files = TRUE,  
  label = NULL  
)
```

Arguments

g a `gtypes` object.
dememorization, **batches**, **iterations** parameters for GENEPOP MCMC LD procedure as defined in `test_LD`.
delete.files logical. Delete GENEPOP input and output files when done?
label character string to use to label GENEPOP input and output files.

Value

data.frame of disequilibrium estimates between pairs of loci

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 N_e 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,
  drop.missing = FALSE,
  num.cores = 1
)
```

Arguments

<code>g</code>	a gtypes object.
<code>maf.threshold</code>	smallest minimum allele frequency permitted to include a locus in calculation of Ne.
<code>by.strata</code>	apply the ‘maf.threshold’ by strata. If ‘TRUE’ then loci that are below this threshold in any strata will be removed from the calculation of Ne for all strata. Loci below ‘maf.threshold’ within a stratum are always removed for calculations of Ne for that stratum.
<code>ci</code>	central confidence interval.
<code>drop.missing</code>	drop loci with missing genotypes? If ‘FALSE’, a slower procedure is used where individuals with missing genotypes are removed in a pairwise fashion.
<code>num.cores</code>	The number of cores to use to distribute computations over. If set to NULL, the value will be what is reported by detectCores -1.

Value

a data.frame with one row per strata and the following columns:

`stratum` stratum being summarized

`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, simplify = TRUE)
```

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.
simplify	if there is a single locus, return result in a simplified form? If FALSE a list will be returned with one element per locus.

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)

lowFreqSubs(dolph.haps)
```

maf *Minor Allele Frequencies*

Description

Calculate minor allele frequencies for each locus.

Usage

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

Arguments

<code>g</code>	a gtypes object.
<code>by.strata</code>	logical - return results grouped by strata?
<code>maf.within</code>	if <code>by.strata = TRUE</code> , identify minor allele within each strata independently? If <code>FALSE</code> minor allele is identified from all individuals.

Value

A vector or matrix of minor allele frequencies at each locus.

Author(s)

Eric Archer <eric.archer@noaa.gov>

See Also

[alleleFreqs](#)

Examples

```
data(msats.g)

maf(msats.g)

# minor allele identified from all individuals
maf(msats.g, by.strata = TRUE)

# minor allele identified within each strata
maf(msats.g, by.strata = TRUE, maf.within = TRUE)
```

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",
simplify = TRUE
)

```

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	The number of cores to use. If set to NULL, the value will be what is reported by detectCores -1. Passed to MAFFT argument --thread.
opts	character string other options to provide to command line.
simplify	if TRUE, if x is a single sequence, a single DNABin object is returned, otherwise, a list of alignments is returned.

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	<i>Run Maverick</i>
-------------	---------------------

Description

Run Maverick clustering algorithm

Usage

```
maverickRun(  
  g,  
  params = NULL,  
  label = "Maverick_files",  
  data_fname = "data.txt",  
  param_fname = "parameters.txt",  
  exec = "Maverick1.0.5"  
)
```

Arguments

<code>g</code>	a gtypes object.
<code>params</code>	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 (<code>data</code> , <code>headerRow_on</code> , <code>missingData</code> , <code>ploidy</code> , <code>ploidyCol_on</code> , <code>popCol_on</code>), so those will be ignored. For a full list of available parameters and their definitions, see the Maverick documentation distributed with the program.
<code>label</code>	folder where input and output files will be written to.
<code>data_fname</code>	file name of data input file.
<code>param_fname</code>	file name of parameters file.
<code>exec</code>	name of executable for Maverick.

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,
  as.haplotypes = TRUE
)
```

Arguments

<code>file</code>	a MEGA-formatted file of sequences.
<code>g</code>	a gtypes object.
<code>label</code>	label for MEGA filename (.meg). If NULL, the gtypes description is used if present.
<code>line.width</code>	width of sequence lines.
<code>locus</code>	number or name of locus to write.
<code>as.haplotypes</code>	output sequences as haplotypes? If TRUE, contents of @sequences slot are returned - treated as if they were haplotypes. If FALSE, one sequence per individual is returned.

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 are on the edges of the cloud of distances. These are the ones that have the greatest mean pairwise distance and greatest variance in distances.

Usage

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

Arguments

x	a set of sequences or a gtypes object with sequences.
num.seqs	number of sequences to return. If NULL (default), all sequences are returned from most to least distant.
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.
simplify	if there is a single locus, return result in a simplified form? If FALSE a list will be returned with one element per locus.

Value

a vector of the num.seqs sequence names that are the most divergent sorted from greatest to least distant.

Author(s)

Eric Archer <eric.archer@noaa.gov>

Examples

```
data(dolph.haps)

mostDistantSequences(dolph.haps, 5)
```

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,
  as.haplotypes = TRUE,
  simplify = TRUE
)
```

Arguments

<code>x</code>	a DNABin object.
<code>num.seqs</code>	number of sequences to return. If NULL (default), all sequences are returned.
<code>model</code>	a character string specifying the evolutionary model to be used. See dist.dna for more information.
<code>pairwise.deletion</code>	a logical indicating whether to delete sites with missing data. See dist.dna for more information.
<code>as.haplotypes</code>	treat sequences as haplotypes (TRUE) or expand haplotypes to one sequence per individual (FALSE). If the latter, individual frequencies are used in cluster formation.
<code>simplify</code>	if there is a single locus, return result in a simplified form? If FALSE a list will be returned with one element per locus.

Value

a vector of the sequence names.

Author(s)

Eric Archer <eric.archer@noaa.gov>

Examples

```
data(dolph.seqs)
mostRepresentativeSequences(dolph.seqs, 5)
mostRepresentativeSequences(dolph.seqs, 3)
```

mRatio	<i>M ratio</i>
--------	----------------

Description

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

Usage

```
mRatio(g, by.strata = FALSE, 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.

Note

The function will only compute the metric for microsatellite 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 loci that do not have all numeric alleles. NA will also be returned if a locus is monomorphic, the locus has no genotypes, or a minimum repeat size cannot be found for all alleles at a locus.

Author(s)

Eric Archer <eric.archer@noaa.gov>

References

Garza, J.C. and E.G. Williamson. 2001. Detection of reduction in population size using data from 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	<i>Dolphin Microsatellite gtypes Object</i>
---------	---

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	<i>Nei's Da</i>
-------	-----------------

Description

Calculate 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

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

Value

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

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(dloop.g)

nd <- nucleotideDivergence(dloop.g)
nd$within
nd$between
```

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

Vector of nucleotide diversity by site.

Author(s)

Eric Archer <eric.archer@noaa.gov>

Examples

```
data(dloop.g)

nd <- nucleotideDiversity(dloop.g)
quantile(nd)
```

numAlleles	<i>Number of Alleles</i>
------------	--------------------------

Description

Return the number of alleles for each locus.

Usage

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

Arguments

g a [gtypes](#) object.
by.strata logical - return results grouped by strata?

Value

vector of number of alleles per locus.

Author(s)

Eric Archer <eric.archer@noaa.gov>

Examples

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

numGenotyped	<i>Number of Individuals Genotyped</i>
--------------	--

Description

Return the number of individuals genotyped for each locus.

Usage

```
numGenotyped(g, by.strata = FALSE, prop = FALSE)
```

Arguments

g a [gtypes](#) object.
by.strata logical - return results grouped by strata?
prop logical determining whether to return proportion missing.

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, by.strata = FALSE, prop = FALSE)
```

Arguments

`g` a `gtypes` object.

`by.strata` logical - return results grouped by strata?

`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	<i>Permute strata</i>
---------------	-----------------------

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	<i>PHASE</i>
-------	--------------

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

<code>g</code>	a gtypes object.
<code>loci</code>	vector or data.frame of loci in 'g' that are to be phased. If a data.frame, it should have columns named <code>locus</code> (name of locus in 'g'), <code>group</code> (number identifying loci in same linkage group), and <code>position</code> (integer identifying location of each locus in a linkage group).
<code>positions</code>	position along chromosome of each locus.
<code>type</code>	type of each locus.
<code>num.iter</code>	number of PHASE MCMC iterations.
<code>thinning</code>	number of PHASE MCMC iterations to thin by.
<code>burnin</code>	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 of 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)
```

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

```

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

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

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

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

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

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

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

```

```

    ...
  )

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

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

Hstats(g)

```

Arguments

<code>g</code>	a gtypes object.
<code>nrep</code>	number specifying number of permutation replicates to use for permutation test.
<code>keep.null</code>	logical. Keep the null distribution from the permutation test?
<code>...</code>	optional arguments passed to or from other functions.
<code>prime.type</code>	type of G'st to calculate. Can be "hedrick" or "nei".
<code>model, gamma, pairwise.deletion</code>	parameters passed to dist.dna . Note that defaults for these arguments (in particular <code>model</code>) are the same as in <code>dist.dna</code> .

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, statFis** Nei, M. and R.K. Chesser. 1983. Estimation of fixation indices and gene diversities. *Ann. Hum. Genet.* 47:253-259.
- statFst** Weir, B. and Cockerham, C. 1984. Estimating F-Statistics for the Analysis of Population Structure. *Evolution* 38(6):1358-1370. doi:10.2307/2408641
- statGst, statGstPrime** Nei, M. 1973. Analysis of gene diversity in subdivided populations. *Proc. Nat. Acad. Sci.* 70(12):3321-3323. Hedrick, P.W. 2005. A standardized genetic differentiation measure. *Evolution* 59(8):1633-1638.
- statFstPrime, statGstDbIPrime** Miermans, P.G. and P.W. Hedrick. 2011. Assessing population structure: FST and related measures. *Molecular Ecology Resources* 11:5-18.
- statJostD** Jost, L. 2008. GST and its relatives do not measure differentiation. *Molecular Ecology* 17:4015-4026.
- statPHist** Excoffier, L., P.E. Smouse, and J.M. Quattro. 1992. Analysis of molecular variance inferred from metric distances among DNA haplotypes: Application to human mitochondrial DNA restriction data. *Genetics* 131:479-491.

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 = 1,
  write.output = FALSE,
  ...
)
```

```
overallTest(
  g,
  nrep = 1000,
  stats = "all",
  keep.null = FALSE,
  quietly = FALSE,
  max.cores = 1,
  ...
)
```



```

)

pairwiseTest(
  g,
  nrep = 1000,
  stats = "all",
  keep.null = FALSE,
  quietly = FALSE,
  max.cores = 1,
  ...
)

```

Arguments

<code>g</code>	a gtypes object.
<code>nrep</code>	number specifying number of permutation replicates to use for permutation test.
<code>stats</code>	a character vector or list of functions specifying which analyses 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.
<code>type</code>	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.
<code>keep.null</code>	logical. Keep the null distribution from the permutation test?
<code>quietly</code>	logical. Print progress and results?
<code>max.cores</code>	The maximum number of cores to use to distribute separate statistics over. The number of cores to use to distribute separate statistics over. If set to NULL, the value will be what is reported by detectCores - 1. If detectCores reports NA, <code>max.cores</code> will be set to 1.
<code>write.output</code>	logical. Write a .csv file with results?
<code>...</code>	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 = "gst", 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, by.strata = FALSE)
```

Arguments

`g` a [gtypes](#) object.
`by.strata` logical - return results grouped by strata?

Value

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

Author(s)

Eric Archer <eric.archer@noaa.gov>

See Also

[privateAlleles](#)

Examples

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

readGenData	<i>Read Genetic Data</i>
-------------	--------------------------

Description

A wrapper for [fread](#) 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 fread .
...	other arguments passed to fread .

Value

a data.frame.

Author(s)

Eric Archer <eric.archer@noaa.gov>

removeSequences	<i>Remove Sequences</i>
-----------------	-------------------------

Description

Remove sequences not used by samples.

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	<i>Convert Sequences To gtypes</i>
-----------------	------------------------------------

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 list to carry other related information (optional).

Value

a [gtypes](#) object.

Author(s)

Eric Archer <eric.archer@noaa.gov>

See Also

[gtypes.initialize](#), [as.matrix.gtypes](#), [as.data.frame.gtypes](#), [gtypes2genind](#), [gtypes2loci](#), [gtypes2phyDat](#)

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,
  plot = TRUE,
  simplify = TRUE,
  ...
)
```

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 suppress plotting.
plot	generate a plot of top n most unlikely sequences.
simplify	if there is a single locus, return result in a simplified form? If FALSE a list will be returned with one element per locus.
...	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-likelihoods) = 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

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

sfs

Site Frequency Spectrum

Description

Calculate the SFS from a data frame of SNP genotypes

Usage

```
sfs(
  x,
  strata.col = 2,
  locus.col = 3,
  fsc.dimnames = TRUE,
  sort.strata = TRUE,
  na.action = c("fail", "filter")
)
```

Arguments

x a data frame of SNP genotypes where the first two columns are id and strata designations and SNPs start on the third column. SNP genotypes are coded as integers where 0 and 2 are the major and minor homozygotes and 1 is the heterozygote.

strata.col column number that strata designations are in.

locus.col	column number that loci start in. All columns after this are assumed to be loci.
fsc.dimnames	format matrix dimnames for fastsimcoal2? If TRUE, then row and column names will be prefixed with the deme number (e.g., "d0_") that they represent.
sort.strata	if joint = TRUE, are strata to be sorted alphabetically? If FALSE then strata are taken in the order found in strata.col.
na.action	action to take if genotypes are missing for some samples. If "fail", an error is thrown if any genotypes are missing. If "filter", SNPs with missing genotypes are removed.

Value

A list of the marginal (1D) and joint (2D) site frequency spectra. If only one stratum is present, then \$marginal will be NULL.

Author(s)

Eric Archer <eric.archer@noaa.gov>

sharedLoci

Shared Loci

Description

Calculate proportion of alleles and number of loci shared between pairs of individuals or strata.

Usage

```
propSharedLoci(g, type = c("strata", "ids"))
```

```
sharedAlleles(g, smry = c("num", "which"))
```

Arguments

<code>g</code>	a <code>gtypes</code> object.
<code>type</code>	a character vector determining type of pairwise comparison. Can be "strata" for strata or "ids" for individuals.
<code>smry</code>	a character vector determining type of summary for sharedAlleles. "which" returns the names of the alleles shared. "num" returns the number of alleles shared.

Value

data.frame summary of pairwise shared loci.

Author(s)

Eric Archer <eric.archer@noaa.gov>

Examples

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

sharedAlleles(msats.g)

## Not run:
propSharedLoci(msats.g)

## End(Not run)
```

simGammaHaps	<i>Simulate Haplotypes</i>
--------------	----------------------------

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)
```

strataSplit	<i>Split Strata</i>
-------------	---------------------

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)
lapply(msats.list, propUniqueAlleles)
```

stratify	<i>Stratify gtypes</i>
----------	------------------------

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. If NULL, all individuals are assigned to a single stratum named "Default".
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 they will be used to match with [getIndNames](#) 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

[getSchemes](#)

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 = getNumStrata(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

<code>g</code>	a gtypes object.
<code>k.range</code>	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.
<code>num.k.rep</code>	number of replicates for each value in <code>k.range</code> .
<code>label</code>	label to use for input and output files
<code>delete.files</code>	logical. Delete all files when STRUCTURE is finished?
<code>exec</code>	name of executable for STRUCTURE. Defaults to "structure".
<code>...</code>	arguments to be passed to <code>structureWrite</code> .
<code>maxpops</code>	number of groups.
<code>burnin</code>	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 identifying 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

structureRun a list where each element is a list with results from structureRead and a vector of the filenames used

structureWrite a vector of the filenames used by STRUCTURE

structureRead 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"),
    plot = TRUE
  )

```

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 preferred 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).
plot	display plot?

Value

invisibly, the ggplot object

Author(s)

Eric Archer <eric.archer@noaa.gov>

See Also

[structure](#), [clumpp](#)

`summarizeAll`*Summarize Genotypes and Sequences*

Description

Conducts a suite of data summaries. 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

```
summarizeAll(g, write.files = FALSE, label = NULL, ...)
```

Arguments

<code>g</code>	a gtypes object.
<code>write.files</code>	logical determining whether to write .csv files of summaries
<code>label</code>	optional label for output folder and prefix for files.
<code>...</code>	optional arguments to pass on to summary functions.

Value

If `write.files = TRUE`, 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

[summarizeInds](#), [summarizeLoci](#), [dupGenotypes](#), [lowFreqSubs](#), [sequenceLikelihoods](#)

summarizeInds	<i>Individual Summaries</i>
---------------	-----------------------------

Description

Compile standard by-individual summaries.

Usage

```
summarizeInds(g)
```

Arguments

`g` a [gtypes](#) object.

Value

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

`id` The individual id

`stratum` The stratum of the individual

`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)
```

```
summarizeInds(msats.g)
```

`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.

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)
```

summarizeSeqs	<i>Sequence Summaries</i>
---------------	---------------------------

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	<i>Summarize gtypes Object</i>
------------------------	--------------------------------

Description

Generate a summary of a gtypes object.

Usage

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

Arguments

object a `gtypes` object.
 ... other arguments (ignored).

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, CI = 0.95)
```

Arguments

x set of DNA sequences or a haploid `gtypes` object with sequences.
 CI desired central confidence interval.

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	<i>Theta</i>
-------	--------------

Description

Calculate theta from heterozygosity of each locus.

Usage

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

Arguments

`g` a [gtypes](#) object.
`by.strata` logical - return results grouped by strata?

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	<i>Trim N's From Sequences</i>
--------	--------------------------------

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)
```

variableSites	<i>Variable Sites</i>
---------------	-----------------------

Description

Identify variable sites among sequences.

Usage

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

Arguments

x	a gtypes object with sequences, a DNABin object, or a list of sequences.
bases	character vector of bases to consider.
simplify	if there is a single locus, return result in a simplified form? If FALSE a list will be returned with one element per locus.

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.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>

writeGtypes	<i>Write gtypes</i>
-------------	---------------------

Description

Write a [gtypes](#) object to file(s).

Usage

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

Arguments

<code>g</code>	a gtypes object.
<code>label</code>	label for filename(s). Default is the gtypes description if present.
<code>folder</code>	folder where file(s) should be written to. If NULL, files are written to current working directory.
<code>by.strata</code>	if <code>as.frequency == TRUE</code> , calculate frequencies by strata?
<code>as.frequency</code>	logical indicating if haploid data should be output as frequency tables.
<code>freq.type</code>	if <code>as.frequency == TRUE</code> , write absolute frequencies ("freq") or proportions ("prop").
<code>...</code>	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)  
writeGtypes(msats.g, one.col = TRUE)  
  
# Write control region data as frequency tables  
data(dloop.g)  
writeGtypes(dloop.g, as.frequency = TRUE)  
  
## End(Not run)
```

Index

*Topic **datasets**

- bowhead.snp.position, 12
- bowhead.snps, 13
- dloop.g, 17
- dolph.haps, 18
- dolph.msats, 18
- dolph.seqs, 19
- dolph.strata, 19
- msats.g, 68

*Topic **package**

- strataG-package, 4

[, gtypes, ANY, ANY, ANY-method
(gtypes.accessors), 43

accessors (gtypes.accessors), 43

alleleFreqs, 4, 61

alleleSplit, 5

allelicRichness, 6

arlequin, 7

arlequinRead (arlequin), 7

arlequinWrite (arlequin), 7

arp2gtypes (arlequin), 7

as.data.frame (as.data.frame.gtypes), 9

as.data.frame, gtypes-method

(as.data.frame.gtypes), 9

as.data.frame.gtypes, 9, 17, 47–49, 85

as.matrix, 10, 105

as.matrix (as.data.frame.gtypes), 9

as.matrix, gtypes-method

(as.data.frame.gtypes), 9

as.matrix.gtypes, 49, 85

as.matrix.gtypes

(as.data.frame.gtypes), 9

as.multidna, 11

baseFreqs, 11

bowhead.snp.position, 12

bowhead.snps, 13

clumpp, 13, 21, 94, 95

createConsensus, 15

dA (nucleotideDivergence), 69

detectCores, 34, 39, 59, 62, 81

df2gtypes, 10, 15, 26, 42, 47–49, 53

dgamma, 89

dist.dna, 65, 66, 69, 79, 86

dloop.g, 17

DNABin, 11, 16, 55, 60, 62, 64, 66, 85, 86, 99,
103, 104

dolph.haps, 18

dolph.msats, 18

dolph.seqs, 19

dolph.strata, 19

dupGenotypes, 20, 96

evanno, 21, 94

expandHaplotypes, 22, 56

expectedNumAlleles (popGenEqns), 76

fasta, 23

fixedDifferences, 23

fixedSites, 24, 24, 104

fread, 84

freq2GenData, 25

fsc.input, 26, 32, 35, 37

fscBlock_dna (fsc.input), 26

fscBlock_freq (fsc.input), 26

fscBlock_microsat (fsc.input), 26

fscBlock_snp (fsc.input), 26

fscBlock_standard (fsc.input), 26

fscCleanup (fscRun), 33

fscDeme (fsc.input), 26

fscEstParam (fsc.input), 26

fscEvent (fsc.input), 26

fscRead, 30, 30, 34, 35, 37

fscReadArp (fscRead), 30

fscReadParamEst (fscRead), 30

fscReadSFS (fscRead), 30

fscRun, 30, 32, 33, 36, 37

- fscSettings (fsc.input), 26
- fscSettingsDef, 36
- fscSettingsDef (fsc.input), 26
- fscSettingsDemes, 36
- fscSettingsDemes (fsc.input), 26
- fscSettingsEst, 36
- fscSettingsEst (fsc.input), 26
- fscSettingsEvents, 36
- fscSettingsEvents (fsc.input), 26
- fscSettingsGenetics, 36
- fscSettingsGenetics (fsc.input), 26
- fscSettingsMigration, 36
- fscSettingsMigration (fsc.input), 26
- fscTutorial (fscRun), 33
- fscWrite, 29, 30, 32, 34, 35, 36
- fstToNm (popGenEqns), 76
- fusFs, 38

- gelato, 39
- gelatoPlot (gelato), 39
- genepop, 40, 51, 58
- genepopWrite (genepop), 40
- genind, 47
- genind2gtypes (gtypes2genind), 47
- getAlleleNames (gtypes.accessors), 43
- getAlleleNames, gtypes-method (gtypes.accessors), 43
- getDescription (gtypes.accessors), 43
- getDescription, gtypes-method (gtypes.accessors), 43
- getIndNames, 91
- getIndNames (gtypes.accessors), 43
- getIndNames, gtypes-method (gtypes.accessors), 43
- getLociNames (gtypes.accessors), 43
- getLociNames, gtypes-method (gtypes.accessors), 43
- getNumInd (gtypes.accessors), 43
- getNumInd, gtypes-method (gtypes.accessors), 43
- getNumLoci (gtypes.accessors), 43
- getNumLoci, gtypes-method (gtypes.accessors), 43
- getNumStrata (gtypes.accessors), 43
- getNumStrata, gtypes-method (gtypes.accessors), 43
- getOther, 56
- getOther (gtypes.accessors), 43
- getOther, gtypes-method (gtypes.accessors), 43
- getPloidy (gtypes.accessors), 43
- getPloidy, gtypes-method (gtypes.accessors), 43
- getSchemes, 91
- getSchemes (gtypes.accessors), 43
- getSchemes, gtypes-method (gtypes.accessors), 43
- getSequences, 11, 45
- getSequences (gtypes.accessors), 43
- getSequences, gtypes-method (gtypes.accessors), 43
- getStrata (gtypes.accessors), 43
- getStrata, gtypes-method (gtypes.accessors), 43
- getStrataNames (gtypes.accessors), 43
- getStrataNames, gtypes-method (gtypes.accessors), 43
- gtypes, 5–9, 11, 12, 15–17, 20, 22–24, 38–40, 43, 45, 47–53, 55, 58, 59, 61, 63–65, 67–75, 79, 81–85, 88, 90–92, 96–98, 100–102, 104, 105
- gtypes (gtypes-class), 42
- gtypes-class, 42
- gtypes.accessors, 42, 43
- gtypes.initialize, 17, 42, 85
- gtypes.initialize (initialize, gtypes-method), 52
- gtypes2genind, 17, 47, 48, 49, 85
- gtypes2loci, 17, 47, 48, 49, 85
- gtypes2phyDat, 17, 49, 85

- heterozygosity, 50
- Hstats (popStructStat), 78
- hw.test, 51
- HWE (hweTest), 51
- hwe (hweTest), 51
- hweTest, 41, 51

- index (gtypes.accessors), 43
- initialize, gtypes-method, 52
- initialize.gtypes, 47–49
- initialize.gtypes (initialize, gtypes-method), 52
- is.gtypes, 53
- isTi (TiTvRatio), 102
- isTv (TiTvRatio), 102
- iupac, 54

- iupacCode (iupac), 54
- iupacMat (iupac), 54
- labelHaplotypes, 5, 22, 55
- landscape2df (landscape2gtypes), 57
- landscape2gtypes, 57
- LDgenepop, 41, 57
- ldNe, 58
- loci, 48
- loci2gtypes (gtypes2loci), 48
- lowFreqSubs, 60, 96
- maf, 60
- mafft, 61
- maverickRun, 63
- MEGA (mega), 64
- mega, 64
- mega, (mega), 64
- mostDistantSequences, 65
- mostRepresentativeSequences, 66
- mRatio, 67
- msats.g, 68
- multidna, 11, 16, 42, 45, 52, 53, 85
- neiDa, 68
- new (initialize, gtypes-method), 52
- nucleotideDivergence, 69
- nucleotideDiversity, 70
- numAlleles, 71
- numGenotyped, 71
- numGensEq (popGenEqns), 76
- numMissing, 72
- overallTest (popStructTest), 80
- pairwiseTest (popStructTest), 80
- permuteStrata, 73
- phase, 73
- phaseFilter (phase), 73
- phasePosterior (phase), 73
- phaseReadPair (phase), 73
- phaseReadSample (phase), 73
- phaseWrite (phase), 73
- phyDat, 49
- phyDat2gtypes (gtypes2phyDat), 49
- popGenEqns, 76
- popStructStat, 78, 81
- popStructTest, 79, 80
- privateAlleles, 82, 83
- propSharedLoci (sharedLoci), 88
- propUniqueAlleles, 83, 83
- read.arlequin (arlequin), 7
- read.fasta (fasta), 23
- read.mega (mega), 64
- readGenData, 84
- removeSequences, 84
- sequence2gtypes, 17, 42, 47–49, 53, 85
- sequenceLikelihoods, 86, 96
- setDescription (gtypes.accessors), 43
- setDescription<- (gtypes.accessors), 43
- setDescription<- , gtypes-method (gtypes.accessors), 43
- setOther (gtypes.accessors), 43
- setOther<- (gtypes.accessors), 43
- setOther<- , gtypes-method (gtypes.accessors), 43
- setSchemes (gtypes.accessors), 43
- setSchemes<- (gtypes.accessors), 43
- setSchemes<- , gtypes-method (gtypes.accessors), 43
- setStrata (gtypes.accessors), 43
- setStrata<- (gtypes.accessors), 43
- setStrata<- , gtypes-method (gtypes.accessors), 43
- sfs, 87
- sharedAlleles (sharedLoci), 88
- sharedLoci, 88
- simGammaHaps, 89
- statChi2 (popStructStat), 78
- statFis (popStructStat), 78
- statFst (popStructStat), 78
- statFstPrime (popStructStat), 78
- statGst (popStructStat), 78
- statGstDblPrime (popStructStat), 78
- statGstPrime (popStructStat), 78
- statJostD (popStructStat), 78
- statPhist (popStructStat), 78
- strataG (strataG-package), 4
- strataG-package, 4
- strataSplit, 90
- stratify, 90
- structure, 14, 21, 91, 94, 95
- structurePlot, 94, 94
- structureRead (structure), 91
- structureRun (structure), 91
- structureWrite (structure), 91

subset (gtypes.accessors), 43
subType (TiTvRatio), 102
summarizeAll, 96
summarizeInds, 96, 97
summarizeLoci, 96, 98
summarizeSeqs, 99
summary (summary, gtypes-method), 99
summary, gtypes-method, 99
summary.gtypes (summary, gtypes-method),
99

tajimasD, 100
tempdir, 36
test_HW, 51
test_LD, 58
theta, 101
theta.h, 101
TiTvRatio, 102
trimNs, 103

validIupacCodes, 54
validIupacCodes (iupac), 54
variableSites, 24, 25, 103

wrightFst (popGenEqns), 76
write.arlequin (arlequin), 7
write.dna, 23, 62
write.fasta (fasta), 23
write.mega (mega), 64
write.nexus.snapp, 104
writeGtypes, 105