Package 'dartR'

June 23, 2017

Type Package

Title Importing and Analysing Snp and Silicodart Data Generated by Genome-Wide Restriction Fragment Analysis

Version 0.80

Date 2017-06-24

Description Functions are provided that facilitate the import and analysis of snp and silicodart (presence/absence) data. The main focus is on data generated by DarT (Diversity Arrays Technology). However, once SNP or related fragment presence/absence data from any source is imported into a genlight object many of the functions can be used. Functions are available for input and output of snp and silicodart data, for reporting on and filtering on various criteria (e.g. CallRate, Heterozygosity, Reproducibility, maximum allele frequency). Advanced filtering is based on Linkage Disequilibrium and HWE. Other functions are available for visualization after PCoA, or to facilitate transfer of data between genlight/genind objects and newhybrids, related, phylip packages etc.

VignetteBuilder knitr

Depends R (>= 3.1.1), adegenet (>= 2.0.0)

Imports plyr, tidyr, reshape2, MASS, ggplot2, directlabels, quadprog, rgl, misc3d, pca3d, utils, Demerelate, seqinr, pegas, SNPassoc, methods, doParallel, stats, data.table, parallel, foreach, stringr, ape, vegan, SNPRelate

Suggests knitr, rmarkdown, StAMPP, mmod

License GPL-2

LazyData true

RoxygenNote 6.0.1

NeedsCompilation no

Author Bernd Gruber [aut, cre], Arthur Georges [aut], Peter J. Unmack [ctb], Oliver Berry [ctb]

Maintainer Bernd Gruber <bernd.gruber@canberra.edu.au>

Repository CRAN

Date/Publication 2017-06-23 16:00:50 UTC

R topics documented:

`dart2genlight` *Convert DarT to genlight*

Description

converts a dart file (read via `read.dart`) into an genlight object `adegenet`

Usage

```
dart2genlight(dart, covfilename = NULL, probar = TRUE)
```

Arguments

`dart`	a dart object created via read.dart
`covfilename`	optional file in csv format with covariates for each individual (see details for explanation)
`probar`	show progress bar

Details

the covariate file needs to have very specific headings. First an heading called id. Here the ids have to match the ids in the dart object `colnames(dart[[4]])`. The following column headings are optional. pop: specifies the population membership of each individual. lat and lon specify spatial coordinates (perferable in decimal degrees WGS1984 format). Additional columns with individual covariates can be imported (e.g. age, gender).

Value

a genlight object is returned. Including all available slots are filled. loc.names, ind.names, pop, lat, lon (if provided via the covariate file)

Examples

```
## Not run:
dgl <- dart2genlight(dart, "covariates.csv")

## End(Not run)
```

`filter.dart` *Filter function to facilitate analysing of dart data [deprecated use explicit filter functions gl.filter...]*

Description

Filter function to facilitate analysing of dart data [deprecated use explicit filter functions gl.filter...]

Usage

```
filter.dart(...)
```

Arguments

`...`	vectors of true false created via comparisons between metrics (see examples)

Value

returns the combined index of filters. this function is unusual as it has no predefined parameters. The idea is here that you can provide any type of filter (and index of true false that is used to filter the loci of your genlight object afterwards. This functions combines your filter and provides a plot on their single and combined effect. see example

Examples

```
## Not run:
#gl is a genlight object created with the read.dart and dart2genlight functions
index.repro <- gl@other$loc.metrics[,"RepAvg"] > 0.98
index.callrate <-  gl@other$loc.metrics[,"CallRate"] > 0.90
index.highhet <- fox.gl.keep@other$loc.metrics[,"FreqHets"] <0.75
index.comb <- filter.dart(index.repro, index.callrate, index.coverage)

## End(Not run)
```

gi.hwe.pop — *Filter function to facilitate analysing of dart data*

Description

Filter function to facilitate analysing of dart data

Usage

```
gi.hwe.pop(gi, pvalue = 0.05, plot = TRUE)
```

Arguments

gi	a genind object (often created using the gl2gi function)
pvalue	the p-value for the HWE test.
plot	a switch if a barplot is wanted.

Value

tests HWE for every loci in every population (using the setupSNP function in package SNPassoc

Examples

```
## Not run:
HWEperpop(gi=geni, pvalue = 0.05, plot = TRUE)

## End(Not run)
```

gi.outflank — *Functio to identify loci under selection per population using the outflank method of Whitlock and Lotterhos (2015)*

Description

Functio to identify loci under selection per population using the outflank method of Whitlock and Lotterhos (2015)

Usage

```
gi.outflank(gi, plot = TRUE)
```

Arguments

gi	a genind object, with a defined population structure
plot	a switch if a barplot is wanted.

Details

this function is a wrapper around the outflank function provided by Whitlock and Lotterhus. To be able to run this function the packages qvalue (from bioconductor) and outflank (from github) needs to be installed. To do so see example below.

Value

returns an index of outliers and the full outflank list

References

Whitlock, M.C. and Lotterhos K.J. (2015) Reliable detection of loci responsible for local adaptation: inference of a neutral model through trimming the distribution of Fst. The American Naturalist 186: 24 - 36.

Examples

```
## Not run:
#install qvalue from bioconductor and outflank from github
source("http://bioconductor.org/biocLite.R")
biocLite("qvalue")
library(devtools)
install_github("Whitlock/OutFLANK")
# now we are able to run outflank using a genlight object
gi.outflank(gi, plot = TRUE)

## End(Not run)
```

---

gi.report.ld *Calculates pairwise population based Linkage Disequilibirum across all loci using the specifyied number of cores*

Description

this function is implemented in a parallel fashion to speed up the process. There is also the ability to restart the function if crashed by specifying the chunkfile names or restarting the function exactly in the same way as in the first run. This is implemented as sometimes due to connectivity loss between cores the function my crash half way.

Usage

```
gi.report.ld(gi, name = NULL, save = TRUE, nchunks = 2, ncores = 1,
  chunkname = NULL)
```

Arguments

gi	a genind object created via gl2gi
name	character string for rdata file. If not given genind object name is used
save	switch if results are saved in a file
nchunks	how many subchunks will be used (the less the faster, but if the routine crashes more bits are lost
ncores	how many cores should be used
chunkname	the name of the chunks for saving, default is NULL

Value

returns calculation of pairwise LD across all loci between subpopulation. This functions uses if specified many cores on your computer to speed up. And if save is used can restart (if save=TRUE is used) with the same command starting where it crashed.

Author(s)

Bernd Gruber (glbugs@aerg.canberra.edu.au)

---

gi2gl *Converts a genind object to genlight object*

Description

Converts a genind object to genlight object

Usage

```
gi2gl(gi)
```

Arguments

gi	– a genind object

Details

Be aware due to ambiguity which one is the reference allele a combination of gi2gl(gl2gi(gl)) does not return an identical object (but in terms of analysis this conversions are equivalent)

Value

A genlight object, with all slots filled.

Author(s)

Bernd Gruber (glbugs@aerg.canberra.edu.au)

`gl.collapse` *Collapse a distance matrix by amalgamating populations*

Description

This script takes a distance matrix (d in lower matrix) and generates a population recode table to amalgamate populations with distance less than or equal to a specified threshold The distance matrix is generated by gl.fixed.diff()

Usage

```
gl.collapse(fd, gl, recode.table = "tmp.csv", t = 0, iter = 1)
```

Arguments

`fd`	– name of the distance matrix produced by gl.fixed.diff() [required]
`gl`	– name of the genlight object from which the distance matrix was calculated [required]
`recode.table`	– name of the new recode.table to receive the population reassignments arising from the amalgamation of populations [default tmp.csv]
`t`	– the threshold distance value for amalgamating populations [default 0]
`iter`	– a parameter to indicate the cycle when gl.collapse() is used interatively [default 1]

Value

The new genlight object with recoded populations

Author(s)

Arthur Georges and Aaron Adamnack (glbugs@aerg.canberra.edu.au)

Examples

```
#only used the first 20 individuals due to runtime reasons
fd <- gl.fixed.diff(testset.gl[1:20,], t=0.05)
gl <- gl.collapse(fd, testset.gl[1:20,], recode.table="testset_recode.csv",t=0.026)
```

gl.collapse.recursive *Recursively collapse a distance matrix by amalgamating populations*

Description

This script generates a fixed difference matrix from a genlight object {adegenet} and from it generates a population recode table used to amalgamate populations with distance less than or equal to a specified t. The script then repeats the process until there is no further amalgamation of populations. The distance matricies are generated by gl.fixed.diff(), a recode table is generated using gl.collapse() and the resultant recode table is applied to the genlight object using gl.pop.recode(). The process is repeated as many times as necessary to yield a final table with no distances less than or equal to the specified threshold. The intermediate and final recode tables and distance matricies are stored to disk as csv files for use with other analyses. In particular, the recode tables can be editted to replace Group1, Group2 etc with meaninful names.

Usage

```
gl.collapse.recursive(gl, prefix = "collapse", threshold = 0)
```

Arguments

gl	– name of the genlight object from which the distance matricies are to be calculated [required]
prefix	– a string to be used as a prefix in generating the matricies of fixed differences (stored to disk) and the recode tables (also stored to disk) [default "collapse"]
threshold	– the threshold distance value for amalgamating populations [default 0]

Value

The new genlight object with recoded populations.

Author(s)

Arthur Georges (glbugs@aerg.canberra.edu.au)

Examples

```
#only used the first 20 individuals due to runtime reasons
fd <- gl.collapse.recursive(testset.gl[1:20,], prefix="testset",threshold=0.026)
```

gl.dist *Calculate a distance matrix for populations defined in an {adegenet} genlight object*

Description

This script calculates various distances between populations based on allele frequencies.

Usage

```
gl.dist(gl, method = "euclidean", binary = FALSE, diag = TRUE,
  upper = FALSE, p = NULL)
```

Arguments

gl	– name of the genlight containing the SNP genotypes [required]
method	– Specify distance measure [method=euclidean]
binary	– Perform presence/absence standardization before analysis using decostand [binary=FALSE]
diag	– Compute and display diagonal
upper	– Return also upper triangle
p	– The power of the Minkowski distance

Details

The distance measure can be one of "manhattan", "euclidean", "canberra", "bray", "kulczynski", "jaccard", "gower", "altGower", "morisita", "horn", "mountford", "raup" , "binomial", "chao", "cao", "mahalanobis", "maximum", "binary" or "minkowski". Refer to the documentation for dist stats or vegdist vegan for definitions.

Value

A matrix of distances between populations (class dist)

Author(s)

Arthur Georges (glbugs@aerg.canberra.edu.au)

Examples

```
gl.dist(testset.gl, method="gower", diag=TRUE)
```

gl.edit.recode.ind *Create or edit a individual (=specimen) names and create an recode_ind file*

Description

A script to edit individual names in a genlight object, or to create a reassignment table taking the individual labels from a genlight object, or to edit existing individual labels in an existing recode_ind file.

Usage

```
gl.edit.recode.ind(gl, ind.recode = NULL)
```

Arguments

gl	Name of the genlight object for which individuals are to be relabelled.[required]
ind.recode	Name of the file to output the new assignments [optional]

Details

Renaming individuals may be required when there have been errors in labelling arising in the process from sample to DArT files. There may be occasions where renaming individuals is required for preparation of figures. Caution needs to be exercised because of the potential for breaking the "chain of evidence" between the samples themselves and the analyses. REcoding individuals can be done with a recode table (csv).

This script will input an existing recode table for editting and optionally save it as a new table, or if the name of an input table is not supplied, will generate a table using the individual labels in the parent genlight object.

The script, having deleted individuals, identifies resultant monomorphic loci or loci with all values missing and deletes them (using gl.filter.monomorphs.r)

The script returns a genlight object with the new individual labels.

Value

An object of class ("genlight") with the revised individual labels

Author(s)

Arthur Georges (glbugs@aerg.canberra.edu.au)

Examples

```
## Not run:
gl <- gl.edit.recode.ind(gl)
gl <- gl.edit.recode.ind(gl, pop.recode="ind.recode.table.csv")
gl <- gl.edit.recode.ind(gl, pop.recode="ind.recode.table.csv",

## End(Not run)
#Ammended Georges 9-Mar-17
```

---

gl.edit.recode.pop *Create or edit a population re-assignment table*

Description

A script to edit population assignments in a genlight object, or to create a reassignment table taking the population assignments from a genlight object, or to edit existing population assignments in a pop.recode.table.

Usage

```
gl.edit.recode.pop(gl, pop.recode = NULL)
```

Arguments

gl	Name of the genlight object for which populations are to be reassigned.[required]
pop.recode	Name of the file to output the new assignments [optional]

Details

Genlight objects assign specimens to populations based on information in the ind.metadata file provided when the genlight object is first generated. Often one wishes to subset the data by deleting populations or to amalgamate populations. This can be done with a pop.recode table with two columns. The first column is the population assignment in the genlight object, the second column provides the new assignment.

This script will input an existing reassignment table for editting and optionally save it as a new table, or if the name of an input table is not supplied, will generate a table using the population assignments in the parent genlight object.

The script, having deleted populations, identifies resultant monomorphic loci or loci with all values missing and deletes them (using gl.filter.monomorphs.r)

The script returns a genlight object with the new population assignments.

Value

An object of class ("genlight") with the revised population assignments

Author(s)

Arthur Georges (glbugs@aerg.canberra.edu.au)

Examples

```
## Not run:
gl <- gl.edit.recode.pop(gl)
gl <- gl.edit.recode.pop(gl, pop.recode="pop.recode.table.csv")
gl <- gl.edit.recode.pop(gl, pop.recode="pop.recode.table.csv",
pop.recode.new="recode.table.minus.hybrids.csv")

## End(Not run)
```

---

gl.filter.callrate — *Filter loci or specimens in a genlight {adegenet} object based on call rate*

Description

SNP datasets generated by DArT have missing values primarily arising from failure to call a SNP because of a mutation at one or both of the the restriction enzyme recognition sites. This script filters out loci (or specimens) for which the call rate is lower than a specified value. The script will also filter out loci (or specimens) in SilicoDArT (presence/absence) datasets where the call rate is lower than the specified value. In this case, the data are missing owing to low coverage.

Usage

```
gl.filter.callrate(gl, method = "loc", threshold = 0.95)
```

Arguments

`gl`	– name of the genlight object containing the SNP data, or the genind object containing the SilocoDArT data [required]
`method`	– "loc" to specify that loci are to be filtered, "ind" to specify that specimens are to be filtered [default "loc"]
`threshold`	– threshold value below which loci will be removed [default 0.95]

Value

The reduced genlight or genind object, plus a summary

Author(s)

Arthur Georges (glbugs@aerg.canberra.edu.au)

Examples

```
result <- gl.filter.callrate(testset.gl, method="ind", threshold=0.8)
```

gl.filter.cloneid *Filter for CloneID to select only unique SNPs*

Description

Filter for CloneID to select only unique SNPs

Usage

```
gl.filter.cloneid(gl)
```

Arguments

gl	a genlight object created via read.dart (needs to have a cloneID as provided by dart)

Value

filtered genlight object, with unique cloneIDs

Examples

```
{
}
```

gl.filter.dups *Filter duplicated loci in a genlight {adegenet} object*

Description

SNP datasets generated by DArT include fragments with more than one SNP and record them separately with the same CloneID (=AlleleID). These multiple SNP loci within a fragment are likely to be linked, and so you may wish to remove duplicates. This script filters out duplicate loci after ordering the genlight object on based on reproducibility, PIC in that order.

Usage

```
gl.filter.dups(gl)
```

Arguments

gl	– name of the genlight object containing the SNP data, or the genind object containing the SilocoDArT data [required]

Value

The reduced genlight or genind object, plus a summary

Author(s)

Arthur Georges (glbugs@aerg.canberra.edu.au)

Examples

```
gl <- gl.filter.dups(testset.gl)
```

`gl.filter.hamming` *Filters loci in a genlight object based on pairwise Hamming distance between sequence tags*

Description

Hamming distance is calculated as the number of base differences between two sequences which can be expressed as a count or a proportion. Typically, it is calculated between two sequences of equal length. In the context of DArT trimmed sequences, which differ in length but which are anchored to the left by the restriction enzyme recognition sequence, it is sensible to compare the two trimmed sequences starting from immediately after the common recognition sequence and terminating at the last base of the shorter sequence.

Usage

```
gl.filter.hamming(gl = gl, t = 0.2, rs = 5, probar = TRUE)
```

Arguments

`gl`	– genlight object [required]
`t`	– a threshold Hamming distance for filtering loci [default 0.2]
`rs`	– number of bases in the restriction enzyme recognition sequence [default = 4]
`probar`	– switch to output progress bar [default is false]

Details

Hamming distance can be computed by exploiting the fact that the dot product of two binary vectors x and (1-y) counts the corresponding elements that are different between x and y. This approach can also be used for vectors that contain more than two possible values at each position (e.g. A, C, T or G).

If a pair of DNA sequences are of differing length, the longer is truncated.

The algorithm is that of Johann de Jong https://johanndejong.wordpress.com/2015/10/02/faster-hamming-distance-in-r-2/ as implimented in utils.hamming.r

Only one of two loci are retained if their Hamming distance is less that a specified percentage. 5 base differences out of 100 bases is a 20

Value

a genlight object filtered on Hamming distance.

Author(s)

Arthur Georges (glbugs@aerg.canberra.edu.au)

Examples

```
gl <- gl.filter.hamming(testset.gl, t=0.25)
```

---

gl.filter.hwe — *Filters loci that show significant departure from Hardy-Weinberg Equilibrium*

Description

Calculates the probabilities of agreement with H-W equilibrium based on observed frequencies of reference homozygotes, heterozygotes and alternate homozygotes. Uses the exact calculations contained in function prob.hwe() as developed by Wigginton, JE, Cutler, DJ, and Abecasis, GR.

Usage

```
gl.filter.hwe(gl, p = 0.05, basis = "any", bon = TRUE)
```

Arguments

gl	– a genlight object containing the SNP genotypes [Required]
p	– level of significance (per locus) [Default 0.05]
basis	– basis for filtering out loci (any, HWE departure in any one population) [default basis="any"]
bon	– apply bonferroni correction to significance levels for filtering [default TRUE]

Details

Input is a genlight adegenet object containing SNP genotypes (0 homozygous for reference SNP, 1 heterozygous, 2 homozygous for alternate SNP).

Loci are filtered if they show HWE departure in any one population. Note that power to detect departures from HWE is affected by sample size and that effective filtering may require substantial sample sizes (n > 20).

Value

a genlight object with the loci departing significantly from HWE removed

Author(s)

Arthur Georges (glbugs@aerg.canberra.edu.au)

Examples

```
list <- gl.filter.hwe(testset.gl,0.05, bon=TRUE)
```

gl.filter.monomorphs *Remove monomorphic loci, including those with all NAs*

Description

This script deletes monomorphic loci from a genlight {adegenet} object

Usage

```
gl.filter.monomorphs(gl, probar = FALSE)
```

Arguments

gl	– name of the input genlight object [required]
probar	– switch to output progress bar [default is false]

Details

A DArT dataset will not have monomorphic loci, but they can arise when populations are deleted by assignment or by using the delete option in gl.pop.recode(). Retaining monomorphic loci unnecessarily increases the size of the dataset.

Value

A genlight object with monomorphic loci removed

Author(s)

Arthur Georges (glbugs@aerg.canberra.edu.au)

Examples

```
gl <- gl.filter.monomorphs(testset.gl)
```

gl.filter.repavg *Filter loci in a genlight {adegenet} object based on average reproducibility of alleles at a locus*

Description

SNP datasets generated by DArT have in index, RepAvg, generated by reproducing the data independently for 30% of loci. RepAvg is the proportion of alleles that give a reproducible result, averaged over both alleles for each locus.

Usage

```
gl.filter.repavg(gl, t = 1)
```

Arguments

gl	– name of the genlight object containing the SNP data [required]
t	– threshold value below which loci will be removed [default 0.95]

Value

Returns a genlight object retaining loci with a RepAvg less than the specified threshold deleted.

Author(s)

Arthur Georges (glbugs@aerg.canberra.edu.au)

Examples

```
gl.report.repavg(testset.gl)
result <- gl.filter.repavg(testset.gl, t=0.95)
```

---

gl.fixed.diff *Generate a matrix of fixed differences from a genelight or genind object {adegenet}*

Description

This script takes SNP data grouped into populations in a genlight object (DArTSeq) or presence absence data (SilicoDArT) grouped into populations in a genind object and generates a matrix of fixed differences between populations taken pairwise

Usage

```
gl.fixed.diff(gl, t = 0, probar = FALSE)
```

Arguments

gl	– name of the genlight object containing SNP genotypes or a genind object containing presence/absence data [required]
t	– threshold value for tollerance in when a difference is regarded as fixed
probar	– switch to output progress bar [default is false]

Details

A fixed difference at a locus occurs when two populations share no alleles. The challenge with this approach is that when sample sizes are finite, fixed differences will occur through sampling error, compounded when many loci are examined. Simulations suggest that sample sizes of n1=5 and n2=5 is adequate to reduce the probability of [experiment-wide] type 1 error to negligible levels [ploidy=2]. A warning is issued if comparison between two populations involves sample sizes less than 5, taking into account allele drop-out.

Tollerance in the definition of a fixed difference is provided by the t parameter. For example, t=0.05 means that SNP allele frequencies of 95,5 and 5,95 percent will be regarded as fixed.

Value

Matrix of fixed differences

Author(s)

Arthur Georges (glbugs@aerg.canberra.edu.au)

See Also

is.fixed

Examples

```
#only used the first 20 individuals due to runtime reasons
mat <- gl.fixed.diff(testset.gl[1:20,], t=0.05)
mat
```

`gl.gene.freq` *Calculate a statistic for each locus by group An internal function essentially to convey readability to rather contorted R code. It takes as input a genlight {adegenet} object with an index variable (say, population) and calculates the selected statistic for each locus, broken down by the groups defined by the index variable.*

Description

Calculate a statistic for each locus by group

An internal function essentially to convey readability to rather contorted R code. It takes as input a genlight {adegenet} object with an index variable (say, population) and calculates the selected statistic for each locus, broken down by the groups defined by the index variable.

Usage

```
gl.gene.freq(gl, method = pop(gl), stats = "mean")
```

Arguments

gl	– name of the genlight object containing the SNP data [required]
method	– breakdown variable [default pop(x)]
stats	– statistic to calculate: mean [only mean(x)/2 currently implemented]

Value

A matrix, populations (rows) by loci (columns), showing the statistic [mean/2]

Author(s)

Arthur Georges (glbugs@aerg.canberra.edu.au)

Examples

```
result <- gl.gene.freq(testset.gl, method=pop(gl), stat="mean")
```

---

gl.make.recode.ind *Create a proforma recode_ind file for reassigning individual (=specimen) names*

Description

Renaming individuals may be required when there have been errors in labelling arising in the process from sample to DArT files. There may be occasions where renaming individuals is required for preparation of figures. Caution needs to be exercised because of the potential for breaking the "chain of evidence" between the samples themselves and the analyses. REcoding individuals can be done with a recode table (csv).

Usage

```
gl.make.recode.ind(x, outfile = "default_recode_ind.csv")
```

Arguments

x	– name of the genlight object containing the SNP data, or the genind object containing the SilocoDArT data [required]
outfile	– name of the new proforma file [default default_recode_ind.csv]

Details

This script facilitates the construction of a recode table by producing a proforma file with current individual (=specimen) names in two identical columns. Edit the second column to reassign individual names. Use keyword Delete to delete an individual.

Apply the recoding using gl.recode.ind(). Deleting individuals can potentially generate monomorphic loci or loci with all values missing. Clean this up with gl.filter.monomorphic().

Value

A vector containing the new individual names

Author(s)

Arthur Georges (glbugs@aerg.canberra.edu.au)

Examples

```
## Not run:
result <- gl.make.recode.ind(gl, outfile="Emmac_recode_ind.csv")

## End(Not run)
```

---

gl.make.recode.pop *Create a proforma recode_ind_table file for reassigning individual (=specimen) names*

Description

Renaming individuals may be required when there have been errors in labelling arising in the process from sample to DArT files. There may be occasions where renaming individuals is required for preparation of figures. Caution needs to be exercised because of the potential for breaking the "chain of evidence" between the samples themselves and the analyses. REcoding individuals can be done with a recode table (csv).

Usage

```
gl.make.recode.pop(x, outfile = "recode_ind_table.csv")
```

Arguments

x	– name of the genlight object containing the SNP data, or the genind object containing the SilocoDArT data [required]
outfile	– name of the new proforma file [default recode_ind_table.csv]

Details

This script facilitates the construction of a recode table by producing a proforma file with current individual (=specimen) names in two identical columns. Edit the second column to reassign individual names. Use keyword Delete to delete an individual.

Apply the recoding using gl.recode.ind(). Deleting individuals can potentially generate monomorphic loci or loci with all values missing. Clean this up with gl.filter.monomorphic().

Value

A vector containing the new individual names

Author(s)

Arthur Georges (glbugs@aerg.canberra.edu.au)

Examples

```
## Not run:
result <- gl.make.recode.ind(gl, outfile="Emmac_recode_ind.csv")

## End(Not run)
```

---

gl.pcoa *PCoA ordination (glPca)*

Description

This script takes the data on SNP genotypes for individuals and undertakes a Gower PCoa ordination using Euclidean distance and drawing upon data in the original genlight {adegenet} object (entity x attribute matrix). The script is essentially a wrapper for glPca() {adegenet} with default settings apart from setting parallel=FALSE and converting the eigenvalues to percentages.

Usage

```
gl.pcoa(gl, nfactors = 5)
```

Arguments

gl	Name of the genlight object containing the SNP genotypes by specimen and population [required]
nfactors	Number of dimensions to retain in the output file [default 5]

Value

An object of class glPca containing the eigenvalues, factor scores and factor loadings

Author(s)

Arthur Georges (glbugs@aerg.canberra.edu.au)

Examples

```
pcoa <- gl.pcoa(testset.gl, nfactors=3)
```

`gl.pcoa.plot` *Bivariate plot of the results of a PCoA ordination*

Description

This script takes output from the ordination generated by gl.pcoa() and plots the individuals classified by population.

Usage

```
gl.pcoa.plot(glPca, data, scale = FALSE, ellipse = FALSE, p = 0.95,
  labels = "pop", hadjust = 1.5, vadjust = 1, xaxis = 1, yaxis = 2)
```

Arguments

`glPca`	Name of the glPca object containing the factor scores and eigenvalues [required]
`data`	Name of the genlight object containing the SNP genotypes by specimen and population [required]
`scale`	Flag indicating whether or not to scale the x and y axes in proportion to % variation explained [default FALSE]
`ellipse`	Flag to indicate whether or not to display ellipses to encapsulate points for each population [default FALSE]
`p`	Value of the percentile for the ellipse to encapsulate points for each population [default 0.95]
`labels`	– Flag to specify the labels are to be added to the plot. ["none"|"ind"|"pop"|"interactive"|"legend", default = "pop"]
`hadjust`	Horizontal adjustment of label position [default 1.5]
`vadjust`	Vertical adjustment of label position [default 1]
`xaxis`	Identify the x axis from those available in the ordination (xaxis <= nfactors)
`yaxis`	Identify the y axis from those available in the ordination (yaxis <= nfactors)

Details

The factor scores are taken from the output of gl.pcoa() – an object of class glPca – and the population assignments are taken from from the original data file. The specimens are shown in a bivariate plot optionally with adjacent labels and enclosing ellipses. Population labels on the plot are shuffled so as not to overlap (using package {directlabels}). This can be a bit clunky, as the labels may be some distance from the points to which they refer, but it provides the opportunity for moving labels around using graphics software (Adobe Illustrator).

Any pair of axes can be specified from the ordination, provided they are within the range of the nfactors value provided to gl.pcoa(). Axes can be scaled to represent the proportion of variation explained. In any case, the proportion of variation explained by each axis is provided in the axis label.

Points displayed in the ordination can be identified if the option labels="interactive" is chosen, in which case the resultant plot is ggplotly() friendly. Running ggplotyly() with no parameters will replot the data and allow identification of points by moving the mouse over them. Refer to the plotly package for further information. Do not forget to load the library via library(plotly).

Value

A plot of the ordination

Author(s)

Arthur Georges (glbugs@aerg.canberra.edu.au)

Examples

```
gl <- testset.gl
levels(pop(gl))<-c(rep("Coast",5),rep("Cooper",3),rep("Coast",5),
rep("MDB",8),rep("Coast",7),"Em.subglobosa","Em.victoriae")
pcoa<-gl.pcoa(gl,nfactors=5)
gl.pcoa.plot(pcoa, gl, ellipse=TRUE, p=0.99, labels="pop",hadjust=1.5, vadjust=1)
gl.pcoa.plot(pcoa, gl, ellipse=TRUE, p=0.99, labels="pop",hadjust=1.5, vadjust=1, xaxis=1, yaxis=3)
```

---

gl.pcoa.plot.3d *3D interactive plot of the results of a PCoA ordination*

Description

This script takes output from the ordination undertaken using gl.pcoa() and plots the individuals in 3D space. The visualisation can be rotated with the mouse to examine the structure.

Usage

```
gl.pcoa.plot.3d(x, gl, title = "PCoA", xaxis = 1, yaxis = 2, zaxis = 3,
  shape = "sphere", radius = 2, legend = "topright")
```

Arguments

x	– name of the glPca object containing the factor scores and eigenvalues [required]
gl	– name of the genlight object from which the PCoA was generated
title	– a title for the plot [default "PCoA"]
xaxis	– identify the x axis from those available in the ordination (xaxis <= nfactors) [default 1]
yaxis	– identify the y axis from those available in the ordination (yaxis <= nfactors) [default 2]
zaxis	– identify the z axis from those available in the ordination (zaxis <= nfactors) [default 3]

shape	– shape of the points, one of sphere, tetrahaedron or cube [default "sphere"]
radius	– size of the points [default 2]
legend	– one of bottomright, bottom, bottomleft, left, topleft, top, topright, right, center [default "bottom"]

Details

The factor scores are taken from the output of gl.pcoa(), an object of class glPca, and the population assignments from the original data file and plots the specimens in a 3D plot.

Axes can be specified from the ordination, provided they are within the range of the nfactors value provided to gl.pcoa().

This script is essentially a wrapper for function pca3d {pca3d} maintained by January Weiner.

Value

An interactive 3D plot of the ordination in a separate window

Author(s)

Arthur Georges (glbugs@aerg.canberra.edu.au)

Examples

```
pcoa <- gl.pcoa(testset.gl, nfactor=5)
gl.pcoa.plot.3d(pcoa, testset.gl, xaxis=1, yaxis=2, zaxis=3)
```

---

gl.pcoa.pop *PCoA ordination of populations*

Description

This script takes the data on allele frequencies for populations and undertakes a Gower PCoA ordination using a nominated distance measure. It draws population information and calculates gene frequencies by drawing upon data in the original genlight {adegenet} object (entity x attribute matrix). The script is essentially a wrapper for pcoa() {ape}.

Usage

```
gl.pcoa.pop(gl, c = "none", method = "euclidean")
```

Arguments

gl	– name of the genlight object containing the SNP genotypes by specimen and population [required]
c	– Correction methods for negative eigenvalues: \"lingoes\" and \"cailliez\" Refer to {ape} documentation. [default \"none\"]
method	– the distance measure to be used. This must be one of "euclidean", "maximum", "manhattan", "canberra", "binary" or "minkowski". Any unambiguous substring can be given.

Value

An object of class pcoa containing the eigenvalues, factor scores and factor loadings

Author(s)

Arthur Georges (gl.bugs@aerg.canberra.edu.au)

Examples

```
pcoa <- gl.pcoa.pop(testset.gl)
pcoa <- gl.pcoa.pop(testset.gl, c="cailliez", m="minkowski")
```

---

gl.pcoa.scree *Produce a plot of eigenvalues, standardized as percentages, derived from a PCoA*

Description

This script takes output from gl.pcoa() and produces a plot of eigenvalues, expressed as a percentage of the sum of the eigenvalues. An option is provided to only plot those eigenvalues with greater explanatory power than the average for the original variables.

Usage

```
gl.pcoa.scree(x, top = TRUE)
```

Arguments

x	– name of the pcoa file generated by gl.pcoa() [required]
top	– a flag to indicate whether or not plot only those eigenvalues greater in value than the average for the unordinated original variables (top=TRUE) or to plot all eigenvalues (top=FALSE). If top=FALSE, then a reference line showing the average eigenvalue for the unordinated variables is shown. [default TRUE]

Details

A Scree Plot is a plot of the relative value of eigenvalues, usually expressed as a percentage, that informs a decision on how many dimensions carry with them substantial information worthy of examination. In an ordination, such as PCoA, the axes are ordered on the proportion of variation explained, so the first axis explains the most (has the largest eigenvalue), the second explains the next greatest amount, and so on.

Value

The scree plot

Author(s)

Arthur Georges (glbugs@aerg.canberra.edu.au)

Examples

```
pcoa <- gl.pcoa(testset.gl)
gl.pcoa.scree(pcoa)
```

---

gl.percent.freq *Generate percentage allele frequencies by locus and population*

Description

This is a support script, to take SNP data or SilocoDArT presence/absence data grouped into populations in a genelight or genind object {adegenet} and generate a table of allele frequencies for each population and locus

Usage

```
gl.percent.freq(gl)
```

Arguments

gl	– name of the genlight containing the SNP genotypes or genind object containing the presence/absence data [required]

Value

A matrix with allele frequencies (genelight) or presence/absence frequencies (genind) broken down by population and locus

Author(s)

Arthur Georges (glbugs@aerg.canberra.edu.au)

Examples

```
m <-  gl.percent.freq(testset.gl)
m
```

`gl.read.dart` *Import DarT data into R and conver it to a genlight object*

Description

This function is a wrapper function that allows you to convert you dart file into a genlight object in one step. In previous versions you had to use read.dart and then dart2genlight. In case you have individual metadata for each individual/sample you can specify as before in the dart2genlight command the file that combines the data.

Usage

```
gl.read.dart(filename, covfilename = NULL, nas = "-", topskip = NULL,
  stdmetrics = c("AlleleID", "SNP", "SnpPosition", "RepAvg", "CallRate",
  "AvgCountRef", "AvgCountSnp", "FreqHomRef", "FreqHomSnp", "FreqHets",
  "OneRatioSnp"), addmetrics = NULL, lastmetric = "RepAvg", probar = TRUE)
```

Arguments

`filename`	path to file (csv file only currently)
`covfilename`	the name of the file that has entails additional information on individuals. For the require format check
`nas`	a character specifying NAs (default is "-")
`topskip`	a number specifying the number of rows to be skipped. If not provided the number of rows to be skipped are "guessed" by the number of rows with "*" at the beginning.
`stdmetrics`	a vector of column headings that are extracted. AlleleID and its format is compulsory, the rest are needed for filtering.
`addmetrics`	add additional headers/columns by name
`lastmetric`	specifies the last non genetic column (Default is "RepAvg"). Be sure to check if that is true, otherwise the number of individuals will not match. You can also specify the last column by a number.
`probar`	show progress bar

Value

a dart genlight object that contains individuals [if data were provided] and loci meta data [from a DArT report]. The dart genlight object can then be fed into a number of initial screening, export and export functions provided by the package. For some of the function it is necessary to have the metadata that was provided from DArT. Please check the vignette for more information. Additional information can also be found in the help documents for `read.dart` and `dart2genlight`.

Examples

```
{
dartfile <- system.file("extdata","testset_SNPs_2Row.csv", package="dartR")
covfilename <- system.file("extdata","testset_metadata.csv", package="dartR")
gl <- gl.read.dart(dartfile, covfilename = covfilename, probar=TRUE)
}
```

---

gl.read.dart.arthur *Import SNP data from DArT and convert to genlight {agegenet} format (gl)*

Description

DaRT provide the data as a matrix of entities (individual turtles) across the top and attributes (SNP loci) down the side in a format that is unique to DArT. This program reads the data in to adegenet format (genlight) for consistency with other programming activity. The script or the data may require modification as DArT modify their data formats from time to time.

Usage

```
gl.read.dart.arthur(datafile, topskip, nmetavar, nas = "-",
  ind.metafile = NULL)
```

Arguments

`datafile`	– name of csv file containing the DartSeq data in 2-row format (csv) [required]
`topskip`	– number of rows to skip before the header row (containing the specimen identities [required]
`nmetavar`	– number of columns containing the locus metadata (e.g. AlleleID, RepAvg) [required]
`nas`	– missing data character [default "-"]
`ind.metafile`	– name of csv file containing metadata assigned to each entity (individual) [default NULL]

Details

gl.read.dart() opens the data file (csv comma delimited) and skips the first n=topskip lines. The script assumes that the next line contains the entity labels (specimen ids) followed immediately by the SNP data for the first locus. It reads the SNP data into a matrix of 1s and 0s, and inputs the locus metadata and specimen metadata. The locus metadata comprises a series of columns of values for each locus including the essential columns of AlleleID, SNP, SnpPostion and the desirable variables REpAvg and AvgPIC. Refer to documentation provide by DArT for an explanation of these columns.

The specimen metadata provides the opportunity to reassign specimens to populations, and to add other data relevant to the specimen. The key variables are id (specimen identity which must be the same and in the same order as the DArTSeq file, each unique), pop (population assignment), lat

(latitude, optional) and lon (longitude, optional). id, pop, lat, lon are the column headers in the csv file. Other optional columns can be added.

The SNP matrix, locus names (constructed from the AlleleID, SNP and SnpPosition to be unique), locus metadata, specimen names, specimen metadata are combined into a genlight object. Refer to the genlight documentation (Package adegenet) for further details.

Value

An object of class ("genlight") containing the SNP data, and locus and individual metadata

Author(s)

Arthur Georges (glbugs@aerg.canberra.edu.au)

Examples

```
## Not run:
gl <- gl.read.dart(datafile="SNP_DFwt15-1908_scores_2Row.csv", topskip=6,
nmetavar=16, nas="-", ind.metafile="metadata.csv" )

## End(Not run)
```

---

gl.read.silicodart *Import presence/absence data from SilicoDArT and convert to genind {agegenet} format*

Description

DaRT provide the data as a matrix of entities (individual animals) across the top and attributes (P/A of sequenced fragment) down the side in a format that is unique to DArT. This program reads the data in to adegenet format (genind) for consistency with other programming activity. The script may require modification as DArT modify their data formats from time to time.

Usage

```
gl.read.silicodart(datafile, topskip, nmetavar, nas = "-",
  ind.metafile = NULL)
```

Arguments

datafile	– name of csv file containing the SilicoDArT data [required]
topskip	– number of rows to skip before the header row (containing the specimen identities) [required]
nmetavar	– number of columns containing the locus metadata (e.g. CloneID, Reproducibility) [required]
nas	– missing data character [default "-"]
ind.metafile	– name of csv file containing metadata assigned to each entity (individual) [default NULL]

Details

gl.read.silicodart() opens the data file (csv comma delimited) and skips the first n=topskip lines. The script assumes that the next line contains the entity labels (specimen ids) followed immediately by the SNP data for the first locus. It reads the presence/absence data into a matrix of 1s and 0s, and inputs the locus metadata and specimen metadata. The locus metadata comprises a series of columns of values for each locus including the essential columns of CloneID and the desirable variables Reproducibility and PIC. Refer to documentation provide by DArT for an explanation of these columns.

The specimen metadata provides the opportunity to reassign specimens to populations, and to add other data relevant to the specimen. The key variables are id (specimen identity which must be the same and in the same order as the SilicoDArT file, each unique), pop (population assignment), lat (latitude, optional) and lon (longitude, optional). id, pop, lat, lon are the column headers in the csv file. Other optional columns can be added.

The data matrix, locus names (forced to be unique), locus metadata, specimen names, specimen metadata are combined into a genInd object. Refer to the documentation for {adegenet} for further details.

Value

An object of class ("genInd") containing the presence/absence data, and locus and individual metadata

Author(s)

Arthur Georges (glbugs@aerg.canberra.edu.au)

Examples

```
## Not run:
glind <- gl.read.silicodart(datafile="SNP_DFwt15-1908_scores_2Row.csv", topskip=6,
nmetavar=16, nas="-", ind.metafile="metadata.csv" )

## End(Not run)
```

---

gl.recode.ind *Recode individual (=specimen) labels in a genelight or genind object {adegenet}*

Description

This script recodes individual labels and/or deletes individuals from a DaRT genlight SNP file or a SilicoDArT genind file based on information provided in a csv file.

Usage

```
gl.recode.ind(gl, ind.recode)
```

Arguments

gl	– name of the genlight object containing SNP genotypes or a genind object containing presence/absence data [required]
ind.recode	– name of the csv file containing the individual relabelling [required]

Details

Renaming individuals may be required when there have been errors in labelling arising in the process from sample to DArT files. There may be occasions where renaming individuals is required for preparation of figures. Caution needs to be exercised because of the potential for breaking the "chain of evidence" between the samples themselves and the analyses. Recoding individuals can be done with a recode table (csv).

The script, having deleted individuals, identifies resultant monomorphic loci or loci with all values missing and deletes them (using gl.filter.monomorphs.r)

Value

A genlight or genind object with the recoded and reduced data

Author(s)

Arthur Georges (glbugs@aerg.canberra.edu.au)

See Also

gl.filter.monomorphs

Examples

```
## Not run:
   gl <- gl.recode.ind(testset.gl, ind.recode="testset_pop_recode.csv")

## End(Not run)
```

---

gl.recode.pop *Recode population assignments in a genelight or genind object {adegenet}*

Description

This script recodes population assignments and/or deletes populations from a DaRT genlight SNP file or a SilicoDArT genind file based on information provided in a csv file.

Usage

```
gl.recode.pop(gl, pop.recode)
```

Arguments

gl	– name of the genlight object containing SNP genotypes or a genind object containing presence/absence data [required]
pop.recode	– name of the csv file containing the population reassignments [required]

Details

Individuals are assigned to populations based on the specimen metadata data file (csv) used with gl.read.dart() or gl.read.silicodart(). Recoding can be used to amalgamate populations or to selectively delete or retain populations.

The population recode file contains a list of populations in the genelight or genind object as the first column of the csv file, and the new population assignments in the second column of the csv file. The keyword Delete used as a new population assignment will result in the associated specimen being dropped from the dataset.

The script, having deleted populations, identifies resultant monomorphic loci or loci with all values missing and deletes them (using gl.filter.monomorphs.r)

Value

A genlight or genind object with the recoded and reduced data

Author(s)

Arthur Georges (glbugs@aerg.canberra.edu.au)

See Also

gl.filter.monomorphs

Examples

```
## Not run:
   gl <- gl.recode.pop(gl, pop.recode="pop_recode_table_0.csv")
   glind <- gl.recode.pop(glind, pop.recode="pop_recode_table_0.csv")

## End(Not run)
```

---

gl.report.bases *Summary of base pair frequencies*

Description

This script calculates the frequencies of the four bases, and the frequency of transitions and transversions in a DArT genlight object.

Usage

```
gl.report.bases(gl)
```

Arguments

gl	– name of the DArT genlight object [required]

Details

The script checks if trimmed sequences are included in the locus metadata, and if so, tallies up the numbers of A,T,G and C bases. Only the reference state at the SNP locus is counted. Counts of transitions and transversions assume that there is no directionality, that is C>T is the same as T>C, because the reference state is arbitrary.

Value

Matrix containing the percent frequencies of each base (A,C,T,G) and the transition and transversion frequencies.

Author(s)

Arthur Georges (glbugs@aerg.canberra.edu.au)

Examples

```
lst <- gl.report.bases(testset.gl)
lst
```

---

gl.report.callrate *Report summary of Call Rate for loci or individuals*

Description

SNP datasets generated by DArT have missing values primarily arising from failure to call a SNP because of a mutation at one or both of the the restriction enzyme recognition sites. This script reports the number of missing values for each of several percentiles. The script gl.filter.callrate() will filter out the loci with call rates below a specified threshold.

Usage

```
gl.report.callrate(gl, method = "loc")
```

Arguments

gl	– name of the genlight or genind object containing the SNP data [required]
method	specify the type of report by locus (method="loc") or individual (method="ind") [default method="loc"]

Value

Mean call rate by locus (method="loc") or individual (method="ind")

Author(s)

Arthur Georges (glbugs@aerg.canberra.edu.au)

Examples

```
gl.report.callrate(testset.gl)
```

---

gl.report.dups *Report duplicated loci in a genlight {adegenet} object*

Description

SNP datasets generated by DArT include fragments with more than one SNP and record them separately with the same CloneID (=AlleleID). These multiple SNP loci within a fragment are likely to be linked, and so you may wish to remove duplicates. This script reports duplicate loci.

Usage

```
gl.report.dups(gl)
```

Arguments

gl	– name of the genlight object containing the SNP data, or the genind object containing the SilocoDArT data [required]

Value

1

Author(s)

Arthur Georges (glbugs@aerg.canberra.edu.au)

Examples

```
gl.report.dups(testset.gl)
```

`gl.report.hamming` *Calculates the pairwise Hamming distance between DArT trimmed DNA sequences*

Description

Hamming distance is calculated as the number of base differences between two sequences which can be expressed as a count or a proportion. Typically, it is calculated between two sequences of equal length. In the context of DArT trimmed sequences, which differ in length but which are anchored to the left by the restriction enzyme recognition sequence, it is sensible to compare the two trimmed sequences starting from immediately after the common recognition sequence and terminating at the last base of the shorter sequence.

Usage

```
gl.report.hamming(gl, rs = 5)
```

Arguments

`gl`	– genlight object [required]
`rs`	– number of bases in the restriction enzyme recognition sequence [default = 4]

Details

Hamming distance can be computed by exploiting the fact that the dot product of two binary vectors x and (1-y) counts the corresponding elements that are different between x and y. This approach can also be used for vectors that contain more than two possible values at each position (e.g. A, C, T or G).

If a pair of DNA sequences are of differing length, the longer is truncated.

The algorithm is that of Johann de Jong https://johanndejong.wordpress.com/2015/10/02/faster-hamming-distance-in-r-2/ as implimented in utils.hamming.r

Value

Histogram of Hamming distance for the gl object

Author(s)

Arthur Georges (glbugs@aerg.canberra.edu.au)

Examples

```
gl.report.hamming(testset.gl)
```

gl.report.heterozygosity *Reports hetrozygosity*

Description

Calculates the observed heterozygisities by population from a genlight object and plots as a barchart ordered on heterozygosity.

Usage

```
gl.report.heterozygosity(gl)
```

Arguments

gl	– a genlight object containing the SNP genotypes [Required]

Details

#Input is a genlight adegenet object containing SNP genotypes (0 homozygous for reference SNP, #1 heterozygous, 2 homozygous for alternate SNP).

Value

a dataframe containing population labels, observed heterozygosities and sample sizes

Author(s)

Bernd Gruber (glbugs@aerg.canberra.edu.au)

Examples

```
gl.report.heterozygosity(testset.gl)
```

gl.report.hwe *Reports departure from Hardy-Weinberg Equilibrium*

Description

Calculates the probabilities of agreement with H-W equilibrium based on observed frequencies of reference homozygotes, heterozygotes and alternate homozygotes. Uses the exact calculations contained in function prob.hwe() as developed by Wigginton, JE, Cutler, DJ, and Abecasis, GR.

Usage

```
gl.report.hwe(gl, p = 0.05, subset = "each")
```

Arguments

gl	– a genlight object containing the SNP genotypes [Required]
p	– level of significance (per locus) [Default 0.05]
subset	– list populations to combine in the analysis | each | all [Default "all"]

Details

#Input is a genlight adegenet object containing SNP genotypes (0 homozygous for reference SNP, #1 heterozygous, 2 homozygous for alternate SNP).

Tests are applied to all populations pooled (subset="all"), to each population treated separately (subset="each") or to selected populations (subset=c("pop1","pop2")). Tests for Hwe are only valid if there is no population substructure, and the tests have sufficient power only when there is sufficient sample size (n>20).

Value

a dataframe containing loci, counts of reference SNP homozygotes, heterozygotes and alternate SNP homozygotes; probability of departure from H-W equilibrium, and per locus significance with and without Bonferroni Correction.

Author(s)

Arthur Georges (glbugs@aerg.canberra.edu.au)

Examples

```
list <- gl.report.hwe(testset.gl,0.05, subset=c("EmmacMaclGeor", "EmmacCoopCully"))
list <- gl.report.hwe(testset.gl,0.05, subset="all")
list <- gl.report.hwe(testset.gl,0.05, subset="each")
```

---

gl.report.monomorphs *Report monomorphic loci, including those with all NAs*

Description

This script reports the number of monomorphic loci from a genlight {adegenet} object

Usage

```
gl.report.monomorphs(gl)
```

Arguments

gl	– name of the input genlight object [required]

Details

A DArT dataset will not have monomorphic loci, but they can arise when populations or individuals are deleted. Retaining monomorphic loci unnecessarily increases the size of the dataset.

Value

A report on loci, polymorphic, monomorphic, all NAs

Author(s)

Arthur Georges (glbugs@aerg.canberra.edu.au)

Examples

```
gl <- gl.report.monomorphs(testset.gl)
```

---

gl.report.repavg *Report summary of RepAvg, reproducibility averaged over both alleles for each locus in a genlight adegenet object*

Description

SNP datasets generated by DArT have in index, RepAvg, generated by reproducing the data independently for 30 RepAvg is the proportion of alleles that give a reproducible result, averaged over both alleles for each locus.

Usage

```
gl.report.repavg(gl)
```

Arguments

gl	– name of the genlight object containing the SNP data [required]

Value

– the mean call rate

Author(s)

Arthur Georges (glbugs@aerg.canberra.edu.au)

Examples

```
gl.report.repavg(testset.gl)
```

gl.subsample.loci *Subsample n loci from a genlight object and return as a genlight object*

Description

This is a support script, to subsample a genlight {adegenet} object based on loci. Two methods are used to subsample, random and based on information content (avgPIC)

Usage

```
gl.subsample.loci(gl, n, method = "random")
```

Arguments

gl	– name of the genlight object containing the SNP genotypes by specimen and population [required]
n	– number of loci to include in the subsample [required]
method	– "random", in which case the loci are sampled at random; or avgPIC, in which case the top n loci ranked on information content (AvgPIC) are chosen [default "random"]

Value

A genlight object with n loci

Author(s)

Arthur Georges (glbugs@aerg.canberra.edu.au)

Examples

```
## Not run:
result <- gl.subsample.loci(gl, n=200, method="avgPIC")

## End(Not run)
```

gl.tree.nj *Output an nj tree to summarize genetic similarity among populations*

Description

This function is a wrapper for the nj{ape} function applied to Euclidian distances calculated from the genlight object.

Usage

```
gl.tree.nj(gl, type = "phylogram", outgroup = NULL, labelsize = 0.7)
```

Arguments

gl	– Name of the genlight object containing the SNP data or a genind object containing presence absence data [required]
type	– Type of dendrogram phylogram|cladogram|fan|unrooted [Default Phylogram]
outgroup	– Vector containing the population names that are the outgroups [Default NULL]
labelsize	– Size of the labels as a proportion of the graphics default [Default 0.7]

Value

A tree file of type phylo

Author(s)

Arthur Georges (glbugs@aerg.canberra.edu.au)

Examples

```
gl.tree.nj(testset.gl,type="fan")
```

---

gl.write.csv *Write out data from a gl object adegenet to csv file*

Description

This script writes to file the SNP genotypes with specimens as entities (columns) and loci as attributes (rows). Each row has associated locus metadata. Each column, with header of specimen id, has population in the first row.

Usage

```
gl.write.csv(gl, outfile = NULL)
```

Arguments

gl	– name of the genlight object containing SNP genotypes [required]
outfile	– name of the csv file to write the data to [required]

Details

The data coding differs from the DArT 1 row format in that 0 = reference homozygous, 2 = alternate homozygous, 1 = heterozyous, and NA = missing SNP assignment.

Value

saves a glenlight object to csv

Author(s)

Arthur Georges (glbugs@aerg.canberra.edu.au)

Examples

```
## Not run:
gl.write.csv(gl, outfile="SNP_1row.csv")

## End(Not run)
```

`gl2demerelate` *Create a dataframe suitable for input to package {Demerelate} from a genlight {adegenet} object*

Description

Create a dataframe suitable for input to package {Demerelate} from a genlight {adegenet} object

Usage

```
gl2demerelate(gl)
```

Arguments

`gl`	– name of the genlight object containing the SNP data [required]

Value

A dataframe suitable as input to package {Demerelate}

Author(s)

Arthur Georges (glbugs@aerg.canberra.edu.au)

Examples

```
## Not run:
df <- gl2demerelate(gl)

## End(Not run)
```

`gl2fasta` *Concatenates DArT trimmed sequences and outputs a fastA file.*

Description

Concatenated sequence tags are useful for phylogenetic methods where information on base frequencies and transition and transversion ratios are required (for example, Maximum Liklihood methods). Where relevant, heterozygous loci are resolved before concatenation by either assigning ambiguity codes or by random allele assignment.

Usage

```
gl2fasta(gl, method = 1, outfile = "output.fasta")
```

Arguments

`gl`	– name of the DArT genlight object [required]
`method`	– 1 | 2 | 3 | 4. Type method=0 for a list of options [method=1]
`outfile`	– name of the output file (fasta format) [output.fasta]

Details

Four methods are employed

Method 1 – heterozygous positions are replaced by the standard ambiguity codes. The resultant sequence fragments are concatenated across loci to generate a single combined sequence to be used in subsequent ML phylogenetic analyses.

Method=2 – the heterozyous state is resolved by randomly assigning one or the other SNP variant to the individual. The resultant sequence fragments are concatenated across loci to generate a single composite haplotype to be used in subsequent ML phylogenetic analyses.

Method 3 – heterozygous positions are replaced by the standard ambiguity codes. The resultant SNP bases are concatenated across loci to generate a single combined sequence to be used in subsequent MP phylogenetic analyses.

Method=4 – the heterozyous state is resolved by randomly assigning one or the other SNP variant to the individual. The resultant SNP bases are concatenated across loci to generate a single composite haplotype to be used in subsequent MP phylogenetic analyses.

Trimmed sequences for which the SNP has been trimmed out, rarely, by adaptor mis-identity are deleted.

The script writes out the composite haplotypes for each individual as a fastA file. Requires 'TrimmedSequence' and 'SNP' (position and type of transition/transversion of a locus) to be among the locus metrics (`@other$loc.metrics`) headers.

Value

A new gl object with all loci rendered homozygous

Author(s)

Bernd Gruber and Arthur Georges (glbugs@aerg.canberra.edu.au)

Examples

```
gl <- gl.filter.repavg(testset.gl,t=1)
gl <- gl.filter.callrate(testset.gl,t=.98)
gl2fasta(gl, method=1, outfile="test.fasta")
```

---

gl2faststructure — *Export DArT genlight object {adegenet} to faststructure format (to run faststructure elsewhere)*

Description

Recodes in the quite specific faststructure format (e.g first six columns need to be there, but are ignored...check faststructure documentation (if you find any :-( )))

Usage

```
gl2faststructure(gl, outfile = "gl.str", probar = TRUE)
```

Arguments

gl	– genlight object
outfile	– filename of the output fastA file [default genlight.fasta]
probar	switch to show/hide progress bar

Details

The script writes out the a file in faststructure format.

Author(s)

Bernd Gruber (glbugs@aerg.canberra.edu.au)

gl2gds *Convert a genlight object to gds format SNPRelate*

Description

Package SNPRelate relies on a bit-level representation of a SNP dataset that competes with {adegenet} genlight objects and associated files. This function saves a genlight object to a gds format file.

Usage

```
gl2gds(gl, outfile = "gl2gds.gds")
```

Arguments

gl	– name of the genlight object containing the SNP data [required]
outfile	– file name of the output file (including extension).

Author(s)

Arthur Georges (glbugs@aerg.canberra.edu.au)

Examples

```
gl2gds(testset.gl)
```

gl2gi *Converts a genlight object to genind object*

Description

Converts a genlight object to genind object

Usage

```
gl2gi(gl, probar = TRUE)
```

Arguments

gl	– a genlight object
probar	– switch on/off progress bar

Details

this function uses a faster version of df2genind (from the adgegenet package)

Value

A genind object, with all slots filled.

Author(s)

Bernd Gruber (glbugs@aerg.canberra.edu.au)

`gl2nhyb` *Create an input file for the program NewHybrids*

Description

This function compares two sets of parental populations to identify loci that exhibit a fixed difference, returns an genlight object with the reduced data, and creates an input file for the program NewHybrids using the top 200 loci. In the absence of two identified parental populations, the script will select a random set 200 loci only (method=random) or the first 200 loci ranked on information content (AvgPIC).

Usage

```
gl2nhyb(gl, outfile = "nhyb.txt", p0 = NULL, p1 = NULL, t = 0,
  m = "random")
```

Arguments

`gl`	– name of the genlight object containing the SNP data [required]
`outfile`	– name of the file to become the input file for NewHybrids [default nhyb.txt]
`p0`	– list of populations to be regarded as parental population 0 [default NULL]
`p1`	– list of populations to be regarded as parental population 1 [default NULL]
`t`	– sets the level at which a gene frequency difference is considered to be fixed [default 0]
`m`	– specifies the method to use to select 200 loci for NewHybrids [default random]

Details

A fixed difference occurs when a SNP allele is present in all individuals of one population and absent in the other. There is provision for setting a level of tollerance, e.g. t = 0.05 which considers alleles present at greater than 95 a fixed difference. Only the 200 loci are retained, because of limitations of NewHybids.

It is important to stringently filter the data on RepAvg and CallRate if using the random option. One might elect to repeat the analysis using the random option and combine the resultant posterior probabilites should 200 loci be considered insufficient.

Value

The reduced genlight object

Author(s)

Arthur Georges (glbugs@aerg.canberra.edu.au)

Examples

```
## Not run:
m <- gl2nhyb(gl, c("Pop1", "Pop4"), c("Pop7", "Pop9"), t=0, m="random")

## End(Not run)
```

---

gl2phylip — *Create a Phylip input distance matrix from a genlight (SNP) or genind (SilicoDarT) {adegenet} object*

Description

This function calculates and returns a matrix of Euclidean distances between populations and produces an input file for the phylogenetic program Phylip (Joe Felsenstein).

Usage

```
gl2phylip(gl, outfile = "phyinput.txt", bstrap = 1)
```

Arguments

gl	Name of the genlight object containing the SNP data or a genind object containing presence absence data [required]
outfile	Name of the file to become the input file for phylip [default phyinput.txt]
bstrap	Number of bootstrap replicates [default 1]

Value

Matrix of Euclidean distances between populations

Author(s)

Arthur Georges (glbugs@aerg.canberra.edu.au)

Examples

```
## Not run:
result <- gl2phylip(testset.gl, outfile="test.txt", 1000)

## End(Not run)
```

`is.fixed` *Test to see if two populations are fixed at a given locus*

Description

This script compares two percent allele frequencies and reports TRUE if they represent a fixed difference, FALSE otherwise.

Usage

```
is.fixed(s1, s2, t = 0)
```

Arguments

`s1`	– percentage SNP allele frequency for the first population [required]
`s2`	– percentage SNP allele frequency for the second population [required]
`t`	– threshold value for tollerance in when a difference is regarded as fixed [default 0]

Details

A fixed difference at a locus occurs when two populations share no alleles, noting that SNPs are biallelic (ploidy=2). Tollerance in the definition of a fixed difference is provided by the t parameter. For example, t=0.05 means that SNP allele frequencies of 95,5 and 5,95 percent will be reported as fixed (TRUE).

Value

TRUE (fixed difference) or FALSE (alleles shared) or NA (one or both s1 or s2 missing)

Author(s)

Arthur Georges (glbugs@aerg.canberra.edu.au)

See Also

`gl.fixed.diff`

Examples

```
is.fixed(0,100)
is.fixed(100,0)
is.fixed(80,0)
is.fixed(100,NA)
is.fixed(0,NA)
is.fixed(NA,0)
is.fixed(NaN,100)
```

```
is.fixed(NaN,0)
is.fixed(100,NaN)
is.fixed(0,NaN)
is.fixed(NaN,NaN)
is.fixed(NA,NA)
is.fixed(s1=100, s2=0, t=0)
is.fixed(96, 4, t=0.05)
```

---

prob.hwe *Exact SNP test of Hardy-Weinberg Equilibrium*

Description

This code calculates an exact probability of departure from Hardy-Weinberg Equilibrium as described in Wigginton, JE, Cutler, DJ, and Abecasis, GR (2005) A Note on Exact Tests of Hardy-Weinberg Equilibrium. American Journal of Human Genetics. 76:887-893.

Usage

```
prob.hwe(obs_hets, obs_hom1, obs_hom2)
```

Arguments

obs_hets	– count of heterozygotes by locus
obs_hom1	– count of homozygotes, reference state
obs_hom2	– count of homozygotes, alternate state

Details

Note: return code of -1.0 signals an error condition; return code of NA signals that all alleles are NA for a locus

Value

Exact probability of agreement with HWE

Author(s)

Arthur Georges (glbugs@aerg.canberra.edu.au)

Examples

```
hets <- 20
hom_1 <- 5
hom_2 <- 30
p_value <- prob.hwe(hets, hom_1, hom_2)
```

read.dart *Import DarT data to R*

Description

Import DarT data to R

Usage

```
read.dart(filename, nas = "-", topskip = NULL, stdmetrics = c("AlleleID",
  "SNP", "SnpPosition", "RepAvg", "CallRate", "AvgCountRef", "AvgCountSnp",
  "FreqHomRef", "FreqHomSnp", "FreqHets", "OneRatioSnp"), addmetrics = NULL,
  lastmetric = "RepAvg")
```

Arguments

filename	path to file (csv file only currently)
nas	a character specifying NAs (default is "-")
topskip	a number specifying the number of rows to be skipped. If not provided the number of rows to be skipped are "guessed" by the number of rows with "*" at the beginning.
stdmetrics	a vector of column headings that are extracted. AlleleID and its format is compulsory, the rest are needed for filtering.
addmetrics	add additional headers/columns by name
lastmetric	specifies the last non genetic column (Default is "RepAvg"). Be sure to check if that is true, otherwise the number of individuals will not match. You can also specify the last column by a number.

Value

a list of length 5. #dart format (one or two rows) #individuals, #snps, #non genetic metrics, #genetic data (still two line format, rows=snps, columns=individuals)

Examples

```
{
dartfile <- system.file("extdata","testset_SNPs_2Row.csv", package="dartR")
dart <-read.dart(dartfile)
}
```

testset.gl *A genlight object created via the read.dart functions*

Description

A genlight object created via the read.dart functions

Usage

```
testset.gl
```

Format

genlight object

Author(s)

Arthur Georges <(glbugs@aerg.canberra.edu.au>

testset_metadata *Metadata file. Can be integrated via the dart2genlight function.*

Description

Metadata file. Can be integrated via the dart2genlight function.

Format

csv

Author(s)

Arthur Georges <(glbugs@aerg.canberra.edu.au>

testset_pop_recode *Recode file to be used with the function.*

Description

This test data set is provided to show a typical recode file format.

Format

csv

Author(s)

Arthur Georges <(glbugs@aerg.canberra.edu.au>

testset_SNPs_2Row *Testfile in DArT format (as provided by DArT)*

Description

this test data set is provided to show a typical DArT file format. Can be used to create a genlight object using the read.dart function.

Format

csv

Author(s)

Arthur Georges <(glbugs@aerg.canberra.edu.au>

utils.hamming *Calculates the Hamming distance between two DArT trimmed DNA sequences*

Description

Hamming distance is calculated as the number of base differences between two sequences which can be expressed as a count or a proportion. Typically, it is calculated between two sequences of equal length. In the context of DArT trimmed sequences, which differ in length but which are anchored to the left by the restriction enzyme recognition sequence, it is sensible to compare the two trimmed sequences starting from immediately after the common recognition sequence and terminating at the last base of the shorter sequence.

Usage

```
utils.hamming(str1, str2, r = 4)
```

Arguments

str1	– string containing the first sequence [required]
str2	– string containing the second sequence [required]
r	– number of bases in the restriction enzyme recognition sequence [default = 4]

Details

The Hamming distance between the rows of a matrix can be computed quickly by exploiting the fact that the dot product of two binary vectors x and (1-y) counts the corresponding elements that are different between x and y. This matrix multiplication can also be used for matrices with more than two possible values, and different types of elements, such as DNA sequences.

The function calculates the Hamming distance between all columns of a matrix X, or two matrices X and Y. Again matrix multiplication is used, this time for counting, between two columns x and y, the number of cases in which corresponding elements have the same value (e.g. A, C, G or T). This counting is done for each of the possible values individually, while iteratively adding the results. The end result of the iterative adding is the sum of all corresponding elements that are the same, i.e. the inverse of the Hamming distance. Therefore, the last step is to subtract this end result H from the maximum possible distance, which is the number of rows of matrix X.

If the two DNA sequences are of differing length, the longer is truncated. The initial common restriction enzyme recognition sequence is ignored.

The algorithm is that of Johann de Jong https://johanndejong.wordpress.com/2015/10/02/faster-hamming-distance-in-r-2/

Value

Hamming distance between the two strings

Author(s)

Arthur Georges (glbugs@aerg.canberra.edu.au)

Examples

```
str1 <- "aatgGCTAG"
str2 <- "aatgGCTcG"
utils.hamming(str1,str2)
```

`utils.hwe` *Calculates departure from Hardy-Weinberg Equilibrium. Utility script not meant for end users.*

Description

Calculates the probabilities of agreement with H-W equilibrium based on observed frequencies of reference homozygotes, heterozygotes and alternate homozygotes. Uses the exact calculations contained in function prob.hwe() as developed by Wigginton, JE, Cutler, DJ, and Abecasis, GR.

Usage

```
utils.hwe(x, prob = 0.05)
```

Arguments

`x`	– a genlight object containting the SNP profiles for a population [Required]
`prob`	– level of significance [Default 0.05]

Value

Locus, Hom_1, Het, Hom_2, N, Prob, Sig, BonSig)

Author(s)

Arthur Georges (glbugs@aerg.canberra.edu.au)

Index