Package 'dartR'

February 7, 2019

Type Package

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

Version 1.1.11

Date 2019-02-07

Description Functions are provided that facilitate the import and analysis of SNP (single nucleotide polymorphism) 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 (Hardy-Weinberg equilibrium). Other functions are available for visualization after PCoA (Principle Coordinate Analysis), or to facilitate transfer of data between genlight/genind objects and newhybrids, related, phylip, structure, faststructure packages.

VignetteBuilder knitr

Encoding UTF-8

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

biocViews

Imports plyr, tidyr, reshape2, MASS, ggplot2, directlabels, pca3d, utils, seqinr, pegas, SNPassoc, methods, doParallel, stats, data.table, parallel, foreach, stringr, ape, vegan, SNPRelate, StAMPP, dismo, qvalue, sp, rgdal, igraph, rrBLUP, leaflet, mmod, PopGenReport, gdistance, hierfstat

Suggests knitr, rmarkdown, rgl, plotly

License GPL-2

LazyData true

RoxygenNote 6.1.1

NeedsCompilation no

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

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

Repository CRAN

Date/Publication 2019-02-07 14:13:23 UTC

R topics documented:

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

Description

This a test data set to test the validity of functions within dartR and is based on a DArT SNp data set of simulated bandicoots across Australia. It contains 96 individuals and 1000 SNPs.

Usage

```
bandicoot.gl
```

Format

genlight object

Author(s)

Bernd Gruber (bugs? Post to https://groups.google.com/d/forum/dartr

---

`dart2genlight` *Convert DarT to genlight*

Description

converts a dart file (read via `read.dart`) into an genlight object `adegenet`. Internal function called by gl.read.dart

Usage

```
dart2genlight(dart, ind.metafile = NULL, covfilename = NULL,
  probar = TRUE)
```

Arguments

`dart`	a dart object created via read.dart
`ind.metafile`	optional file in csv format with covariates for each individual (see details for explanation)
`covfilename`	depreciated, use parameter ind.metafile
`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)

`gi2gl` *Converts a genind object to genlight object*

Description

Converts a genind object to genlight object

Usage

```
gi2gl(gi, parallel = TRUE)
```

Arguments

`gi`	– a genind object
`parallel`	– switch to deactivate parallel version. Default set to TRUE. Only for testing purpose.

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 (bugs? Post to https://groups.google.com/d/forum/dartr)

`gl.alf` *Calculates allele frequency of the first and second allele for each loci #' A very simple function to report allele frequencies*

Description

Calculates allele frequency of the first and second allele for each loci #' A very simple function to report allele frequencies

Usage

```
gl.alf(gl)
```

Arguments

`gl`	– a genlight object

Value

a simple data.frame with alf1, alf2

Author(s)

Bernd Gruber (bugs? Post to https://groups.google.com/d/forum/dartr)

`gl.amova` *Performs and AMOVA using genlight data.*

Description

This script performs an AMOVA based on the genetic distance matrix from stamppNeisD() [package StAMPP] using the amova() function from the package PEGAS for exploring within and between population variation. For detailed information use their help pages: ?pegas::amova, ?StAMPP::stamppAmova

Usage

```
gl.amova(x, nperm = 100)
```

Arguments

`x`	– name of the genlight containing the SNP genotypes, with population information [required]
`nperm`	– number of permuations to perform for hypothesis testing [default 100]. Please note should be set to 1000 for analysis.]

Value

An object of class "amova" which is a list with a table of sums of square deviations (SSD), mean square deviations (MSD), and the number of degrees of freedom, and a vector of variance components.

Author(s)

Bernd Gruber (bugs? Post to https://groups.google.com/d/forum/dartr)

gl.assign *Assign an individual of unknown provenance to population*

Description

This script assigns an individual of unknown provenance to one or more target populations based on first, an analysis of private alleles, and then, if the assignment remains ambigous, on the basis of a weighted likelihood index.

Usage

```
gl.assign(x, id, nmin = 10, dim = NULL, alpha = 0.05,
  threshold = 0, v = 3)
```

Arguments

x	– name of the input genlight object [required]
id	– identity label of the focal individual whose provenance is unknown [required]
nmin	– minimum sample size for a target population to be included in the analysis [default 10]
dim	– number of dimensions to retain in the dimension reduction [default k, number of populations]
alpha	– probability level for bounding ellipses in the PCoA plot [default 0.05]
threshold	– populations to retain for consideration; those for which the focal individual has less than or equal to threshold loci with private alleles [default 0]
v	– verbosity: 0, silent or errors only; 1, begin and end; 2, progress log; 3, progress and results; 5, full report [default 3]

Details

The algorithm first identifies those target populations for which the individual has no private alleles. If no single population emerges from this analysis, or if a higher threshold than 0 is chosen for the number of tollerable private alleles, then the following process is followed. (a) The space defined by the loci is ordinated to yield a series of orthogonal axes (independent), a necessary condition for combining likelihoods calculated from each axis. (b) A workable subset of dimensions is chosen, normally equal to the number of target populations or the number of dimensions with substantive eigenvalues, whichever is the smaller. (c) The log-likelihood of the value for the unknown on each axis is calculated, weighted by the eigenvalue for that axis, and summed over all dimensions as an assignment index. The assignment index is calculated for a point on the boundary of the 95

There are three considerations to the assignment. First, consider only those populations for which the unknown has no private alleles. Private alleles are an indication that the unknown does not belong to a target population (provided that the sample size is adequate, say >=10).

Second, consider the PCoA plot for populations where no private alleles have been detected and the position of the unknown in relation to the confidence ellipses. Note, this is considering only the top two dimensions of the ordination, and so an unknown lying outside the confidence ellipse

can be interpreted as it lying outside the confidence envelope. However, if the unknown lies inside the confidence ellipse in two dimensions, then it may still lie outside the confidence envelope. This is good for eliminating populations from consideration, but does not provide confidence in assignment.

Third, consider the assignment probabilities. This approach calculates the squared Generalised Linear Distance (Mahalanobis distance) of the unknown from the centroid for each population, and calculates the probability associated with its quantile under the zero truncated normal distribution. This index takes into account position of the unknown in relation to the confidence envelope in all selected dimensions of the ordination.

Each of these approaches provides evidence, none are 100

Value

A genlight object containing the focal individual (assigned to population "unknown") and #' populations for which the focal individual is not distinctive (number of loci with private alleles less than or equal to thresold t.

Author(s)

Arthur Georges (bugs? Post to https://groups.google.com/d/forum/dartr)

Examples

```
# Test run with a focal individual from the Macleay River (EmmacMaclGeor)
x <- gl.assign(testset.gl, id="UC_00146", nmin=10, alpha=0.05, threshold=1)
```

---

gl.basic.stats *Calculates basic statistics for each loci (Hs, Ho, Fis etc.)*

Description

Based on function basic.stats. Check ?basic.stats for help.

Usage

```
gl.basic.stats(gl, digits = 4)
```

Arguments

gl	– a genlight object
digits	– number of digits that should be returned

Value

several tables and lists with all basic stats. Check basic.stats for details.

Author(s)

Bernd Gruber (bugs? Post to https://groups.google.com/d/forum/dartr)

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

Description

This script takes a the file generated by gl.fixed.diff 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, recode.table = "tmp.csv", tpop = 0, tloc = 0,
  v = 2)
```

Arguments

`fd`	– name of the list of matricies produced by gl.fixed.diff() [required]
`recode.table`	– name of the new recode.table to receive the new population reassignments arising from the amalgamation of populations [default tmp.csv]
`tpop`	– max number of fixed differences used amalgamating populations [default 0]
`tloc`	– threshold defining a fixed difference (e.g. 0.05 implies 95:5 vs 5:95 is fixed) [default 0]
`v`	– verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log ; 3, progress and results summary; 5, full report [default 2]

Details

The script then applies the recode to the genlight object and recalculates the distance and associated matricies.

Value

A list containing the gl object x and the following square matricies [[1]] $gl – the new genlight object with populations collapsed; [[2]] $fd – raw fixed differences; [[3]] $pcfd – percent fixed differences; [[4]] $nobs – mean no. of individuals used in each comparison; [[5]] $nloc – total number of loci used in each comparison; [[6]] $expobs – if test=TRUE, the expected count of false positives for each comparison [by simulation]; [[7]] $prob – if test=TRUE, the significance of the count of fixed differences [by simulation])

Author(s)

Arthur Georges (Post to https://groups.google.com/d/forum/dartr)

Examples

```
fd <- gl.fixed.diff(testset.gl, tloc=0.05)
gl <- gl.collapse(fd, recode.table="testset_recode.csv",tpop=1)
```

`gl.collapse.pval` *Collapse a fixed distance matrix by amalgamating populations for which fixed differences are not significant*

Description

This script takes the output from gl.collapse recursive and further collapses the fixed difference matrix based on the pvalue associated with each comparison. The results are subsets of populations (OTUs) for which diagnosability is non-significant. A recode table is generated applied to the genlight object to reflect the resultant OTUs.

Usage

```
gl.collapse.pval(fd, prefix = "fd.sig", delta = 0.02, reps = 1000,
  alpha = 0.05, v = 2)
```

Arguments

`fd`	– name of the list of matricies output by gl.collapse.recursive run with test=TRUE [required]
`prefix`	– a string to be used as a prefix in generating the matrix of fixed differences (stored to disk) and the recode table (also stored to disk) [default "fd_sig"]
`delta`	Default 0.02
`reps`	number of repetition. Default 1000.
`alpha`	– significance level for test of false positives [default 0.05]
`v`	– verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log ; 3, progress and results summary; 5, full report [default 2]

Details

The recode table and final distance matrix are stored to disk as csv files.

Value

A list containing the gl object x and the following square matricies [[1]] $gl – the input genlight object; [[2]] $fd – raw fixed differences; [[3]] $pcfd – percent fixed differences; [[4]] $nobs – mean no. of individuals used in each comparison; [[5]] $nloc – total number of loci used in each comparison; [[6]] $expobs – if test=TRUE, the expected count of false positives for each comparison [by simulation], otherwise NAs [[7]] $prob – if test=TRUE, the significance of the count of fixed differences [by simulation], otherwise NAs

Author(s)

Arthur Georges (bugs? Post to https://groups.google.com/d/forum/dartr)

`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 a fixed difference count less than or equal to a specified threshold, tpop. The script then repeats the process until there is no further amalgamation of populations.

Usage

```
gl.collapse.recursive(x, prefix = "collapse", tloc = 0, tpop = 1,
  test = TRUE, alpha = 0.05, delta = 0.02, reps = 1000, v = 2)
```

Arguments

`x`	– 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"]
`tloc`	– threshold defining a fixed difference (e.g. 0.05 implies 95:5 vs 5:95 is fixed) [default 0]
`tpop`	– max number of fixed differences allowed in amalgamating populations [default 0]
`test`	– if TRUE, calculate p values for the observed fixed differences [default FALSE]
`alpha`	– significance level for test of false positives [default 0.05]
`delta`	– threshold value for the population minor allele frequency (MAF) from which resultant sample fixed differences are considered true positives [default 0.02]
`reps`	– number of replications to undertake in the simulation to estimate probability of false positives [default 1000]
`v`	– verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log ; 3, progress and results summary; 5, full report [default 2]

Details

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.recode.pop(). The process is repeated as many times as necessary to yield a final table with no fixed differences less than or equal to the specified threshold, tpop.

Optionally, if test=TRUE, the script will test the fixed differences between final OTUs for statistical significance, using simulation, and then further amalgamate populations that for which there are no significant fixed differences at a specified level of significance (alpha). To avoid conflation of true fixed differences with false positives in the simulations, it is necessary to decide a threshold value (delta) for extreme true allele frequencies that will be considered fixed for practical purposes. That is, fixed differences in the sample set will be considered to be positives (not false positives) if they arise from true allele frequencies of less than 1-delta in one or both populations. The parameter delta is typically set to be small (e.g. delta = 0.02).

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 edited to replace populaton labels with meaninful names and reapplied in sequence.

Value

A list containing the gl object x and the following square matricies [[1]] $gl – the input genlight object; [[2]] $fd – raw fixed differences; [[3]] $pcfd – percent fixed differences; [[4]] $nobs – mean no. of individuals used in each comparison; [[5]] $nloc – total number of loci used in each comparison; [[6]] $expobs – if test=TRUE, the expected count of false positives for each comparison [by simulation], otherwise NAs [[7]] $prob – if test=TRUE, the significance of the count of fixed differences [by simulation], otherwise NAs

Author(s)

Arthur Georges (Post to https://groups.google.com/d/forum/dartr)

Examples

```
fd <- gl.collapse.recursive(testset.gl, prefix="testset", test=TRUE, tloc=0, tpop=2, v=2)
```

`gl.costdistances` *Calculates cost distances for a given landscape (resistance matrix)*

Description

calculates a cost distance matrix, to be used with run.popgensim

Usage

```
gl.costdistances(landscape, locs, method, NN)
```

Arguments

landscape	a raster object coding the resistance of the landscape
locs	coordinates of the subpopulations. If a genlight object is provided coordinates are taken from @other$latlong and centers for population (pop(gl)) are calculated. In case you want to calculate costdistances between individuals redefine pop(gl) via: `pop(gl)<- indNames(gl)`.
method	defines the type of cost distance, types are "least-cost", "rSPDistance" or "commute (Circuitscape type)"
NN	number of next neighbours recommendation is 8

Value

a costdistance matrix between all pairs of locs

---

gl.define.pop — *Define a new population in a genelight {adegenet} object on the basis of specified individuals*

Description

The script reassigns existing individuals to a new population and removes their existing population assignment

Usage

```
gl.define.pop(x, ind.list, new, v = 2)
```

Arguments

x	– name of the genlight object containing SNP genotypes [required]
ind.list	– a list of individuals to be assigned to the new population [required]
new	– name of the new population
v	– verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log ; 3, progress and results summary; 5, full report [default 2]

Details

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

Value

A genlight object with the redefined population structure

Author(s)

Arthur Georges (bugs? Post to https://groups.google.com/d/forum/dartr)

Examples

```
gl <- gl.define.pop(testset.gl, ind.list=c("AA019073","AA004859"), new="newguys")
```

`gl.dist.heatmap` *Represent a distance matrix as a heatmap*

Description

The script plots a heat map to represent the distances in the distance or dissimilarity matrix

Usage

```
gl.dist.heatmap(dst, ncolors = 5, labels = TRUE, values = TRUE,
  rank = FALSE, v = 2)
```

Arguments

`dst`	– name of the distance matrix [required]
`ncolors`	– number of colors to display [default 5]
`labels`	– if TRUE, and the number of rows is <= 20, labels are added to the heatmap [default = TRUE]
`values`	– if TRUE, and the number of rows is <= 20, distances are added to the body of the heatmap [default = TRUE]
`rank`	– if TRUE, then the distance matrix will be reordered to group like with like, otherwise order will be displayed as given [default FALSE]
`v`	– verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log ; 3, progress and results summary; 5, full report [default 2]

Author(s)

Arthur Georges (Post to https://groups.google.com/d/forum/dartr)

Examples

```
gl <- testset.gl[1:10,]
d <- dist(as.matrix((gl)))
gl.dist.heatmap(d)
gl.dist.heatmap(d, ncolors=10, rank=TRUE)
```

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

Description

This script calculates various distances between populations based on allele frequencies. The distances are calculated by scripts in the stats or vegan libraries, with the exception of the pcfixed (percent fixed differences) and pa (total private alleles) distances.

Usage

```
gl.dist.pop(x, method = "euclidean", binary = FALSE, diag = TRUE,
  upper = FALSE, p = NULL, v = 2)
```

Arguments

`x`	– 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 [TRUE]
`upper`	– Return also upper triangle [FALSE]
`p`	– The power of the Minkowski distance (typically a value ranging from 0.25 to infinity) [0.5]
`v`	– verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log ; 3, progress and results summary; 5, full report [default 2]

Details

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

Distance pcfixed calculates the pair-wise count of fixed allelic differences between populations. Distance pa tallies the total number of private alleles possessed by one or the other population.

Value

A matrix of distances between populations (class dist)

Author(s)

Arthur Georges (bugs? Post to https://groups.google.com/d/forum/dartr)

Examples

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

`gl.diversity` *Calculate diversity indices for SNPs*

Description

!!Just an intro placeholder!! This script takes a genlight object and calculates alpha and beta diversity for q=0:2. Formulas are taken from Sherwin et al. 2017. The paper describes nicely the relationship between the different q levels and how they relate to population genetic processes such as dispersal and selection. For all indices the entropies (H) and corrosponding effective numbers Hill numbers (D), which reflect the amount of entities that are needed to get the observed valuea are calculated. In a nutshell the alpha indices between the different q-values should be similar if there are no deviation from expected allele frequencies and occurrences (e.g. all loci in HWE & equilibrium). If there is a deviation of an index this links to a process causing it such as dispersal, selection or strong drift. For a detailed explanation of all the indices, we recommend to resort to the literature provided below.

Usage

```
gl.diversity(gl, spectrumplot = TRUE, confiplot = FALSE,
  probar = TRUE, table = "DH")
```

Arguments

`gl`	genlight object containing the SNP genotypes [required]
`spectrumplot`	switch to provide a plot [TRUE]
`confiplot`	switch if confidence intervals (1 sd) should be drawn [default: FALSE]
`probar`	report on progress. Silent if set to FALSE. [Default is TRUE]
`table`	prints a tabular output to the console either 'D'=D values, or 'H'=H values or 'DH','HD'=both or 'N'=no table.

Value

a list of entropy indices for each level of q and equivalent numbers for alpha and beta diversity.

Author(s)

Bernd Gruber (bugs? Post to https://groups.google.com/d/forum/dartr), Contributors: William B. Sherwin, Alexander Sentinella

References

Sherwin, W.B., Chao, A., Johst, L., Smouse, P.E. (2017). Information Theory Broadens the Spectrum of Molecular Ecology and Evolution. TREE 32(12) 948-963. doi:10.1016/j.tree.2017.09.12

Chao et al. 2014

gl.drop.ind *Remove specified individuals from a genelight {adegenet} object*

Description

The script, having deleted individuals, optionally identifies resultant monomorphic loci or loci with all values missing and deletes them (using gl.filter.monomorphs.r). The script also optionally recalculates statistics made redundant by the deletion of individuals from the dataset.

Usage

```
gl.drop.ind(x, ind.list, recalc = FALSE, mono.rm = TRUE, v = 2)
```

Arguments

x	– name of the genlight object containing SNP genotypes or a genind object containing presence/absence data [required]
ind.list	– a list of individuals to be removed [required]
recalc	– Recalculate the locus metadata statistics [default FALSE]
mono.rm	– Remove monomorphic loci [default TRUE]
v	– verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log ; 3, progress and results summary; 5, full report [default 2]

Details

The script returns a genlight object with the individuals deleted and, optionally, the recalculated locus metadata.

Value

A genlight object with the reduced data

Author(s)

Arthur Georges (bugs? Post to https://groups.google.com/d/forum/dartr)

See Also

gl.filter.monomorphs

gl.recalc.metrics

Examples

```
gl <- gl.drop.ind(testset.gl, ind.list=c("AA019073","AA004859"))
```

gl.drop.loc *Remove specified loci from a genelight {adegenet} object*

Description

This script deletes selected loci from the nominated dataset.

Usage

```
gl.drop.loc(x, loc.list, v = 2)
```

Arguments

x	– name of the genlight object containing SNP genotypes or a genind object containing presence/absence data [required]
loc.list	– vector of loci names to be droped.
v	– verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log ; 3, progress and results summary; 5, full report [default 2]

Value

A genlight object with the reduced data

Author(s)

Arthur Georges (bugs? Post to https://groups.google.com/d/forum/dartr)

Examples

```
gl <- gl.drop.loc(testset.gl, loc.list=c("100049687|12-A/G","100050106|50-G/A"))
```

gl.drop.pop *Remove specified populations from a genelight {adegenet} object*

Description

Individuals are assigned to populations based on the specimen metadata data file (csv) used with gl.read.dart().

Usage

```
gl.drop.pop(x, pop.list, recalc = FALSE, mono.rm = TRUE, v = 2)
```

Arguments

x	– name of the genlight object containing SNP genotypes or a genind object containing presence/absence data [required]
pop.list	– a list of populations to be removed [required]
recalc	– Recalculate the locus metadata statistics [default FALSE]
mono.rm	– Remove monomorphic loci [default TRUE]
v	– verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log ; 3, progress and results summary; 5, full report [default 2]

Details

The script, having deleted populations, optionally identifies resultant monomorphic loci or loci with all values missing and deletes them (using gl.filter.monomorphs.r). The script also optionally recalculates statistics made redundant by the deletion of individuals from the dataset.

The script returns a genlight object with the new population assignments and the recalculated locus metadata.

Value

A genlight object with the reduced data

Author(s)

Arthur Georges (bugs? Post to https://groups.google.com/d/forum/dartr)

See Also

gl.filter.monomorphs

gl.recalc.metrics

Examples

```
gl <- gl.drop.pop(testset.gl, pop.list=c("EmsubRopeMata","EmvicVictJasp"))
```

---

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, recalc = TRUE,
  mono.rm = TRUE, v = 1)
```

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]
recalc	– Recalculate the locus metadata statistics [default TRUE]
mono.rm	– Remove monomorphic loci [default TRUE]
v	– v=0, silent; v=1, low verbosity; v=2, high verbosity [default 1]

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 also 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, optionally identifies resultant monomorphic loci or loci with all values missing and deletes them (using gl.filter.monomorphs.r). The script also optionally recalculates statistics made redundant by the deletion of individuals from the dataset.

The script returns a genlight object with the new individual labels and the recalculated locus metadata.

Value

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

Author(s)

Arthur Georges (Post to https://groups.google.com/d/forum/dartr)

Examples

```
## Not run:
gl <- gl.edit.recode.ind(testset.gl)
gl <- gl.edit.recode.ind(testset.gl, ind.recode="ind.recode.table.csv")
gl <- gl.edit.recode.ind(testset.gl, ind.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, recalc = FALSE,
  mono.rm = TRUE, v = 1)
```

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]
`recalc`	– Recalculate the locus metadata statistics if any individuals are deleted [default TRUE]
`mono.rm`	– Remove monomorphic loci [default TRUE]
`v`	– verbosity: 0, silent; 1, brief; 2, verbose [default 1]

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, optionally identifies resultant monomorphic loci or loci with all values missing and deletes them (using gl.filter.monomorphs.r). The script also optionally recalculates statistics made redundant by the deletion of individuals from the dataset.

The script returns a genlight object with the new population assignments and the recalculated locus metadata.

Value

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

Author(s)

Arthur Georges (Post to https://groups.google.com/d/forum/dartr)

Examples

```
## Not run:
gl <- gl.edit.recode.pop(testset.gl)

## 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(x, method = "loc", threshold = 0.95,
  mono.rm = TRUE, recalc = FALSE, plot = FALSE, v = 2)
```

Arguments

`x`	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]
`mono.rm`	– Remove monomorphic loci [default TRUE]
`recalc`	– Recalculate the locus metadata statistics if any individuals are deleted in the filtering [default FALSE]
`plot`	specify if a histogram of call rate is to be produced [default FALSE]
`v`	– verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log ; 3, progress and results summary; 5, full report [default 2]

Details

Because this filter operates on call rate, and previously applied functions may not have recalculated locus metrics, this function recalculates Call Rate before filtering. Recalculaton after filtering remains optional, with no recalculation as the default.

Note that when filtering individuals on call rate, the initial call rate is calculated and compared against the threshold. After filtering, if mono.rm=TRUE, the removal of monomorphic loci will alter the call rates. Some individuals with a call rate initially greater than the nominated threshold, and so retained, may come to have a call rate lower than the threshold. If this is a problem, repeated iterations of this function will resolve the issue.

Value

The reduced genlight or genind object, plus a summary

Author(s)

Arthur Georges (Post to https://groups.google.com/d/forum/dartr)

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.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(x = gl, threshold = 0.2, rs = 5, pb = FALSE,
  v = 2)
```

Arguments

`x`	– genlight object [required]
`threshold`	– a threshold Hamming distance for filtering loci [default 0.2]
`rs`	– number of bases in the restriction enzyme recognition sequence [default = 4]
`pb`	– switch to output progress bar [default FALSE]
`v`	– verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log ; 3, progress and results summary; 5, full report [default 2]

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 (Post to https://groups.google.com/d/forum/dartr)

Examples

```
gl <- gl.filter.hamming(testset.gl, threshold=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(x, alpha = 0.05, basis = "any", bon = TRUE, v = 2)
```

Arguments

x	– a genlight object containing the SNP genotypes [Required]
alpha	– 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]
v	– verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log ; 3, progress and results summary; 5, full report [default 2]

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 (Post to https://groups.google.com/d/forum/dartr)

Examples

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

`gl.filter.maf` *Filter loci on the basis of minor allele frequency (MAF) in a genlight adegenet object*

Description

This script calculates the minor allele frequency for each locus and updates the locus metadata for FreqHomRef, FreqHomSnp, FreqHets and MAF (if it exists). It then uses the updated metadata for MAF to filter loci.

Usage

```
gl.filter.maf(x, threshold = 0.01, v = 2)
```

Arguments

`x`	– name of the genlight object containing the SNP data [required]
`threshold`	– threshold MAF – loci with a MAF less than the threshold will be removed [default 0.01]
`v`	– verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log ; 3, progress and results summary; 5, full report [default 2]

Details

Note the this filter applies to MAF calculated across all individuals, without regard to population structure. It is a means of removing overall rare alleles. To apply this to single populations, use sepPop and lapply.

Value

The reduced genlight dataset

Author(s)

Arthur Georges (Post to https://groups.google.com/d/forum/dartr)

Examples

```
f <- gl.filter.maf(testset.gl, threshold=0.05)
```

`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(x, v = 2, pb = FALSE)
```

Arguments

x	– name of the input genlight object [required]
v	– verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log ; 3, progress and results summary; 5, full report [default 2]
pb	– display progress bar [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 (Post to https://groups.google.com/d/forum/dartr)

Examples

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

`gl.filter.repavg` *Filter loci in a genlight {adegenet} object based on average repeatability 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 repeatable result, averaged over both alleles for each locus.

Usage

```
gl.filter.repavg(x, threshold = 0.99, v = 2)
```

Arguments

`x`	– name of the genlight object containing the SNP data [required]
`threshold`	– threshold value below which loci will be removed [default 0.99]
`v`	– verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log ; 3, progress and results summary; 5, full report [default 2]

Value

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

Author(s)

Arthur Georges (Post to https://groups.google.com/d/forum/dartr)

Examples

```
gl.report.repavg(testset.gl)
result <- gl.filter.repavg(testset.gl, threshold=0.95, v=3)
```

`gl.filter.secondaries` *Filter loci that represent secondary SNPs 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 (secondaries) are likely to be linked, and so you may wish to remove secondaries. This script filters out loci after ordering the genlight object on based on repeatability, avgPIC in that order (method="best") or at random (method="random")

Usage

```
gl.filter.secondaries(x, method = "random", v = 2)
```

Arguments

x	– name of the genlight object containing the SNP data [required]
method	– method of selecting SNP locus to retain, best or random [random]
v	– verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log ; 3, progress and results summary; 5, full report [default 2]

Value

The reduced genlight, plus a summary

Author(s)

Arthur Georges (Post to https://groups.google.com/d/forum/dartr)

Examples

```
result <- gl.report.secondaries(testset.gl)
result2 <- gl.filter.secondaries(testset.gl)
```

---

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) and generates a matrix of fixed differences between populations taken pairwise

Usage

```
gl.fixed.diff(x, tloc = 0, test = FALSE, delta = 0.02, reps = 1000,
  rm.global.monomorphs = TRUE, pb = TRUE, v = 2)
```

Arguments

x	– name of the genlight object containing SNP genotypes [required]
tloc	– threshold defining a fixed difference (e.g. 0.05 implies 95:5 vs 5:95 is fixed) [default 0]
test	– if TRUE, calculate p values for the observed fixed differences [default FALSE]
delta	– the threshold value for the minor allele frequency to regard the difference between two populations to be fixed [default 0.02]

`reps` – number of replications to undertake in the simulation to estimate probability of false positives [default 1000]

`rm.global.monomorphs` – if TRUE, loci that are monomorphic across all individuals are removed before computation [default TRUE]

`pb` – if TRUE, show a progress bar on time consuming loops [default FALSE]

`v` – verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log ; 3, progress and results summary; 5, full report [default 2]

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. The minimum sample size for scoring fixed differences between two populations can be set with the parameter nlimit.

An absolute fixed difference is as defined above. However, one might wish to score fixed differences at some lower level of allele frequency difference, say where percent allele fequencies are 95,5 and 5,95 rather than 100:0 and 0:100. This adjustment can be done with the tloc parameter. For example, tloc=0.05 means that SNP allele frequencies of 95,5 and 5,95 percent will be regarded as fixed when comparing two populations at a locus.

Value

A list containing the gl object x and the following square matricies [[1]] $gl – the input genlight object;, [[2]] $fd – raw fixed differences;, [[3]] $pcfd – percent fixed differences;, [[4]] $nobs – mean no. of individuals used in each comparison;, [[5]] $nloc – total number of loci used in each comparison;, [[6]] $expobs – if test=TRUE, the expected count of false positives for each comparison [by simulation], [[7]] $prob – if test=TRUE, the significance of the count of fixed differences [by simulation])

Author(s)

Arthur Georges (Post to https://groups.google.com/d/forum/dartr)

See Also

`is.fixed`

Examples

```
fd <- gl.fixed.diff(testset.gl, tloc=0, test=TRUE, delta=0.02, reps=100, v=1 )
```

`gl.fst.pop` *Calculate a pairwise fst values for populations in a genlight object*

Description

This script calculates pairwise fst values based on the implementation in the StAMPP package (?stamppFst). It allows to run bootstrap to estimate probability of fst values to be different from zero. For detailed information please check the help pages (?stamppFst).

Usage

```
gl.fst.pop(x, nboots = 100, percent = 95, nclusters = 1)
```

Arguments

`x`	– name of the genlight containing the SNP genotypes [required]
`nboots`	– number of bootstraps to perform across loci to generate confidence intervals and p-values
`percent`	– the percentile to calculate the confidence interval around [defalut = 95]
`nclusters`	– the number of proccesor threads or cores to use during calculations.

Value

A matrix of distances between populations (class dist), if nboots =1, otherwise a list with Fsts (in a matrix), Pvalues (a matrix of pvalues), Bootstraps results (data frame of all runs). Hint: Use `as.matrix(as.dist(fsts))` if you want to have a squared matrix with symmetric entries returned, instead of a dist object.

Author(s)

Bernd Gruber (bugs? Post to https://groups.google.com/d/forum/dartr)

Examples

```
gl.fst.pop(possums.gl, nboots=1)
```

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), stat = "mean")
```

Arguments

gl	– name of the genlight object containing the SNP data [required]
method	– breakdown variable [default pop(x)]
stat	– 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 (Post to https://groups.google.com/d/forum/dartr)

Examples

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

`gl.genleastcost` *Least-cost path analysis based on a friction matrix*

Description

This function calculates the pairwise distances (Euclidean, cost path distances and genetic distances) of populations using a friction matrix and a spatial genind object. The genind object needs to have coordinates in the same projected coordinate system as the friction matrix. The friction matrixcan be either a single raster of a stack of several layers. If a stack is provided the specified cost distance is calculated for each layer in the stack. The output of this function can be used with the functions `wassermann` or `lgrMMRR` to test for the significance of a layer on the genetic structure.

Usage

```
gl.genleastcost(x, fric.raster, gen.distance, NN = NULL,
  pathtype = "leastcost", plotpath = TRUE, theta = 1)
```

Arguments

`x`	a spatial genind object. see ?popgenreport how to provide coordinates in genind objects
`fric.raster`	a friction matrix
`gen.distance`	specification which genetic distance method should be used to calculate pairwise genetic distances between populations ("D", "Gst.Nei", "Gst.Hedrick") or individuals ("Smouse", "Kosman", "propShared")
`NN`	Number of neighbours used when calculating the cost distance (possible values 4,8 or 16). As the default is NULL a value has to be provided if pathtype='leastcost'. NN=8 is most commonly used. Be aware that linear structures may cause artefacts in the least-cost paths, therefore inspect the actual least-cost paths in the provided output.
`pathtype`	Type of cost distance to be calculated (based on function in the `gdistance` package. Available distances are 'leastcost', 'commute' or 'rSPDistance'. See functions in the gdistance package for futher explanations. If the path type is set to 'leastcost' then paths and also pathlength are returned.
`plotpath`	switch if least cost paths should be plotted (works only if pathtype='leastcost'. Be aware this slows down the computation, but it is recommended to do this to check least cost paths visually.
`theta`	value needed for rSPDistance function. see `rSPDistance` in package `gdistance`.

Value

returns a list that consists of four pairwise distance matrixes (Euclidean, Cost, length of path and genetic) and the actual paths as spatial line objects.

Author(s)

Bernd Gruber (bugs? Post to https://groups.google.com/d/forum/dartr)

References

Cushman, S., Wasserman, T., Landguth, E. and Shirk, A. (2013). Re-Evaluating Causal Modeling with Mantel Tests in Landscape Genetics. Diversity, 5(1), 51-72. Landguth, E. L., Cushman, S. A., Schwartz, M. K., McKelvey, K. S., Murphy, M. and Luikart, G. (2010). Quantifying the lag time to detect barriers in landscape genetics. Molecular ecology, 4179-4191. Wasserman, T. N., Cushman, S. A., Schwartz, M. K. and Wallin, D. O. (2010). Spatial scaling and multi-model inference in landscape genetics: Martes americana in northern Idaho. Landscape Ecology, 25(10), 1601-1612.

See Also

landgenreport, popgenreport, wassermann, lgrMMRR

Examples

```
## Not run:
data(possums.gl)
library(raster)  #needed for that example
landscape.sim <- readRDS(system.file("extdata","landscape.sim.rdata", package="dartR"))
glc <- gl.genleastcost(x=possums.gl,fric.raster=landscape.sim ,
gen.distance = "D", NN=8, pathtype = "leastcost",plotpath = TRUE)
library(PopGenReport)
wassermann(eucl.mat = glc$eucl.mat, cost.mat = glc$cost.mats,  gen.mat = glc$gen.mat)
lgrMMRR(gen.mat = glc$gen.mat, cost.mats = glc$cost.mats,  eucl.mat = glc$eucl.mat)

## End(Not run)
```

---

gl.grm *Calculates the genomic relatedness matrix*

Description

The G matrix is calculated by centering the allele frequency matrix of the second allele by substracting 2 times the allefrequency

Usage

```
gl.grm(gl, plotheatmap = TRUE, return.imputed = FALSE, ...)
```

Arguments

gl	– a genlight object
plotheatmap	– a switch if a heatmap should be shown [Default:TRUE]
return.imputed	switch if loci with imputed data should be returned (see ?A.mat in package rrBLUP)
...	parameters passed to function A.mat from package rrBLUP

Value

a genomic relatedness matrix

Examples

```
gl.grm(bandicoot.gl[1:5,1:10])
```

---

gl.grm.network *Represents a genomic relatedness matrix as a network*

Description

This script takes a G matrix generated by gl.grm() and represents the relationship among the specimens as a network diagram. In order to use this script, a decision is required on a threshold for relatedness to be represented as link in the network, and on the layout used to create the diagram.

Usage

```
gl.grm.network(G, x, method = "fr", node.size = 3,
  node.label = FALSE, node.label.size = 0.7,
  node.label.color = "black", alpha = 0.004,
  title = "Network based on G-matrix of genetic relatedness", v = 3)
```

Arguments

G	– a G relatedness matrix generated by gl.grm [required]
x	– genlight object from which the G matrix was generated [required]
method	– one of fr, kk or drl [Default:fr]
node.size	– size of the symbols for the network nodes [default: 3]
node.label	– TRUE to display node labels [default: FALSE]
node.label.size	– Size of the node labels [default: 0.7]
node.label.color	– color of the text of the node labels [default: "black"]
alpha	– upper threshold to determine which links between nodes to display [default: 0.995]
title	– title for the plot [default: "Network based on G-matrix of genetic relatedness"]
v	– verbosity. If zero silent, max 3.

Details

The threshold for relatedness to be represented as a link in the network is specified as a quantile. Those relatedness measures above the quantile are plotted as links, those below the quantile are not. Often you are looking for relatedness outliers in comparison with the overall relatedness among individuals, so a very conservative quantile is used (e.g. 0.004), but ultimately, this decsion is made as a matter of trial and error. One way to approach this trial and error is to try to achieve a sparse set of links between unrelated 'background' individuals so that the stronger links are preferentially shown.

There are several layouts from which to choose. The most popular are given as options in this script.

fr – Fruchterman, T.M.J. and Reingold, E.M. (1991). Graph Drawing by Force-directed Placement. Software – Practice and Experience 21:1129-1164. kk – Kamada, T. and Kawai, S.: An Algorithm for Drawing General Undirected Graphs. Information Processing Letters 31:7-15, 1989. drl – Martin, S., Brown, W.M., Klavans, R., Boyack, K.W., DrL: Distributed Recursive (Graph) Layout. SAND Reports 2936:1-10, 2008.

colors of node symbols are those of the rainbow.

Author(s)

Arthur Georges (bugs? Post to https://groups.google.com/d/forum/dartr)

Examples

```
#gl.grm.network(G,x)
```

---

gl.Ho *A very simple function to report observed Heterozygosity*

Description

A very simple function to report observed Heterozygosity

Usage

```
gl.Ho(gl)
```

Arguments

gl	– a genlight object

Value

a simple vector wiht Ho for each loci

Author(s)

Bernd Gruber (bugs? Post to https://groups.google.com/d/forum/dartr)

`gl.Hs` *A very simple function to report expected Heterozygosity*

Description

A very simple function to report expected Heterozygosity

Usage

```
gl.Hs(gl)
```

Arguments

`gl`	– a genlight object

Value

a simple vector wiht Ho for each loci

Author(s)

Bernd Gruber (bugs? Post to https://groups.google.com/d/forum/dartr)

`gl.hwe.pop` *Filter function to facilitate analysing of dart data*

Description

Filter function to facilitate analysing of dart data

Usage

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

Arguments

`gi`	a genlight or genind object(genlight is internally converted to genind)
`pvalue`	the p-value for the HWE test.
`plot`	a switch if a barplot is wanted.

Value

This functions performs a HWE test for every population (rows) and loci (columns) and returns a true false matrix. True is reported if the p-value of an HWE-test for a particular loci and population was below the specified threshold (pvalue, default=0.05). The thinking behind this approach is that loci that are not in HWE in several populations have most likely to be treated (e.g. filtered if loci under selection are of interest). If plot=TRUE a barplot on the on the loci and the sum of deviation over all population is returned. Loci that deviate in the majority of populations can be identified via colSums on the resulting matrix.

Examples

```
library(parallel)
gl.hwe.pop(testset.gl, pvalue = 0.05, plot = TRUE)
```

---

gl.ibd *Isolation by distance*

Description

This functions performs an isolation by distance analysis based on a mantel test and also produces an isolation by distance plot. If a genlight object with coordinates is provided) then a Euclidean and genetic distance matrix are calculated (currently. Currently only pairwise Fst between population is implemented. Coordinates are expected as lat long and converted to Google Earth Mercator projection. If coordinates are already projected, set projected=TRUE. If such an object is provided an isolation by distance analysis and plot is performed on log(Euclidean distance) against population based pairwise Fst/1-Fst (see Rousseau's distance measure. Genetics April 1, 1997 vol. 145 no. 4 1219-1228) You can provide also your own genetic and Euclidean distance matrix. The function is based on the code provided by the adegenet tutorial (http://adegenet.r-forge.r-project.org/files/tutorial-basics.pdf), using the functions mantel (package vegan), stamppFst (package StAMPP) and Mercator in package dismo.

Usage

```
gl.ibd(gl = NULL, Dgen = NULL, Dgeo = NULL, projected = FALSE,
  permutations = 999, plot = TRUE)
```

Arguments

gl	genlight object. If provided a standard analysis on Fst/1-Fst and log(distance) is performed
Dgen	genetic distance matrix if no genlight object with coordinates is provided
Dgeo	Euclidean distance matrix if no genlight object is provided
projected	Switch to indicate that coordinates are already projected (not in lat long) and therefore no projection is carried out. Default is FALSE, so it is assumed coordinates are in lat/longs.

permutations	number of permutations in the mantel test
plot	should an isolation by distance plot be returned. Default is plot=TRUE

Value

returns a list of the following components: Dgen (the genetic distance matrix), Dgeo (the Euclidean distance matrix), mantel (the statistics of the mantel test)

Author(s)

Bernd Gruber (bugs? Post to https://groups.google.com/d/forum/dartr)

References

Rousset (1997) Genetic Differentiation and Estimation of Gene Flow from F-Statistics Under Isolation by Distancenetics 145(4), 1219-1228.

See Also

mantel, stamppFst

Examples

```
ibd <- gl.ibd(bandicoot.gl)
ibd <- gl.ibd(bandicoot.gl,plot = FALSE)
```

---

gl.keep.ind — *Remove all but the specified individuals from a genelight {adegenet} object*

Description

The script, having deleted individuals, optionally identifies resultant monomorphic loci or loci with all values missing and deletes them (using gl.filter.monomorphs.r). The script also optionally recalculates statistics made redundant by the deletion of individuals from the dataset.

Usage

```
gl.keep.ind(x, ind.list, recalc = FALSE, mono.rm = TRUE, v = 2)
```

Arguments

x	– name of the genlight object containing SNP genotypes or a genind object containing presence/absence data [required]
ind.list	– a list of individuals to be removed [required]
recalc	– Recalculate the locus metadata statistics [default FALSE]
mono.rm	– Remove monomorphic loci [default TRUE]
v	– verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log ; 3, progress and results summary; 5, full report [default 2]

Details

The script returns a genlight object with the individuals deleted and, optionally, the recalculated locus metadata.

Value

A genlight object with the reduced data

Author(s)

Arthur Georges (bugs? Post to https://groups.google.com/d/forum/dartr)

See Also

gl.filter.monomorphs

gl.recalc.metrics

Examples

```
gl <- gl.keep.ind(testset.gl, ind.list=c("AA019073","AA004859"))
```

---

gl.keep.pop — *Remove all but specified populations from a genelight {adegenet} object*

Description

Individuals are assigned to populations based on the specimen metadata data file (csv) used with gl.read.dart().

Usage

```
gl.keep.pop(x, pop.list, recalc = FALSE, mono.rm = TRUE, v = 2)
```

Arguments

x	– name of the genlight object containing SNP genotypes or a genind object containing presence/absence data [required]
pop.list	– a list of populations to be kept [required]
recalc	– Recalculate the locus metadata statistics [default FALSE]
mono.rm	– Remove monomorphic loci [default TRUE]
v	– verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log ; 3, progress and results summary; 5, full report [default 2]

Details

The script, having deleted the specified populations, optionally identifies resultant monomorphic loci or loci with all values missing and deletes them (using gl.filter.monomorphs.r). The script also optionally recalculates statistics made redundant by the deletion of individuals from the dataset.

The script returns a genlight object with the new population assignments and the recalculated locus metadata.

Value

A genlight object with the reduced data

Author(s)

Arthur Georges (bugs? Post to https://groups.google.com/d/forum/dartr)

See Also

gl.filter.monomorphs

gl.recalc.metrics

Examples

```
gl <- gl.keep.pop(testset.gl, pop.list=c("EmsubRopeMata","EmvicVictJasp"))
```

---

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",
  outpath = tempdir())
```

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]
outpath	– path where to save the output file (set to tempdir by default)

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 (Post to https://groups.google.com/d/forum/dartr)

Examples

```
result <- gl.make.recode.ind(testset.gl, outfile="Emmac_recode_ind.csv")
```

---

gl.make.recode.pop *Create a proforma recode_pop_table file for reassigning population names*

Description

Renaming populations may be required when there have been errors in assignment arising in the process from sample to DArT files or when one wishes to amalgamate populations, or delete populations. Recoding populations can also be done with a recode table (csv).

Usage

```
gl.make.recode.pop(x, outfile = "recode_pop_table.csv",
  outpath = tempdir())
```

Arguments

x	– name of the genlight object containing the SNP data [required]
outfile	– name of the new proforma file [default recode_pop_table.csv]
outpath	– path where to save the output file (set to tempdir by default)

Details

This script facilitates the construction of a recode table by producing a proforma file with current population names in two identical columns. Edit the second column to reassign populations. Use keyword Delete to delete a population.

Apply the recoding using gl.recode.pop().

Value

A vector containing the new population names

Author(s)

Arthur Georges (Post to https://groups.google.com/d/forum/dartr)

Examples

```
result <- gl.make.recode.pop(testset.gl, outfile="Emmac_recode_pop.csv")
```

---

gl.map.interactive *Creates an interactive map (based on latlong) from a genlight object*

Description

Creates an interactive map (based on latlong) from a genlight object

Usage

```
gl.map.interactive(x, provider = "Esri.NatGeoWorldMap")
```

Arguments

x	– a genlight object [including coordinates within the latlong slot]
provider	– passed to leaflet

Details

A wrapper around the **leaflet** package. For possible background maps check as specified via the provider: http://leaflet-extras.github.io/leaflet-providers/preview/index.html

Value

plots a map

Author(s)

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

Examples

```
#gl.map(bandicoot.gl)
```

`gl.merge.pop` *Merge two or more populations in a genelight {adegenet} object into one population*

Description

Individuals are assigned to populations based on the specimen metadata data file (csv) used with gl.read.dart().

Usage

```
gl.merge.pop(x, old = NULL, new = NULL, v = 2)
```

Arguments

`x`	– name of the genlight object containing SNP genotypes or a genind object containing presence/absence data [required]
`old`	– a list of populations to be merged [required]
`new`	– name of the new population [required]
`v`	– verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log ; 3, progress and results summary; 5, full report [default 2]

Details

This script assigns individuals from two nominated populations into a new single population. It can also be used to rename populations.

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

Value

A genlight object with the new population assignments

Author(s)

Arthur Georges (bugs? Post to https://groups.google.com/d/forum/dartr)

Examples

```
gl <- gl.merge.pop(testset.gl, old=c("EmsubRopeMata","EmvicVictJasp"), new="Outgroup")
```

`gl.nhybrids` *Create an input file for the program NewHybrids and run it if NewHybrids is installed*

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

```
gl.nhybrids(gl, outfile = "nhyb.txt", outpath = tempdir(), p0 = NULL,
  p1 = NULL, t = 0, method = "random", nhyb.directory = NULL,
  BurnIn = 10000, sweeps = 10000, GtypFile = "TwoGensGtypFreq.txt",
  AFPriorFile = NULL, PiPrior = "Jeffreys", ThetaPrior = "Jeffreys",
  v = 2)
```

Arguments

`gl`	– name of the genlight object containing the SNP data [required]
`outfile`	– name of the file that will be the input file for NewHybrids [default nhyb.txt]
`outpath`	– path where to save the output file (set to tempdir by default)
`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]
`method`	– specifies the method (random or AvgPIC) to select 200 loci for NewHybrids [default random]
`nhyb.directory`	– directory that holds the NewHybrids executable file e.g. C:/NewHybsPC [default NULL]
`BurnIn`	– number of sweeps to use in the burn in [default 10000]
`sweeps`	– number of sweeps to use in computing the actual Monte Carlo averages [default 10000]
`GtypFile`	– name of a file containing the genotype frequency classes [default TwoGensGtypFreq.txt]
`AFPriorFile`	– name of the file containing prior allele frequency information [default NULL]
`PiPrior`	– Jeffreys-like priors or Uniform priors for the parameter pi [default Jeffreys]
`ThetaPrior`	– Jeffreys-like priors or Uniform priors for the parameter theta [default Jeffreys]
`v`	– verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log ; 3, progress and results summary; 5, full report [default 2]

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.

If you specify a directory for the NewHybrids executable file, then the script will create the input file from the snp data then run NewHybrids. If the directory is set to NULL, the exectution will stop once the input file (nhyb.txt) has been written to disk.

Refer to the New Hybrids manual for further information on the parameters to set – http://ib.berkeley.edu/labs/slatkin/eriq/soft

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

Value

The reduced genlight object, if parentals are provided; output of NewHybrids to disk

Author(s)

Arthur Georges (bugs? Post to https://groups.google.com/d/forum/dartr)

Examples

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

m <- gl.nhybrids(testset.gl, outfile="nhyb.txt",
p0=NULL, p1=NULL,
nhyb.directory="C:/workspace/R_analysis/NewHybsPC",
BurnIn=100,
sweeps=1000,
v=3)

## End(Not run)
```

---

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

Description

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

Usage

```
gl.outflank(gi, plot = TRUE, LeftTrimFraction = 0.05,
  RightTrimFraction = 0.05, Hmin = 0.1, qthreshold = 0.05, ...)
```

Arguments

gi	a genlight of genind object, with a defined population structure
plot	a switch if a barplot is wanted.
LeftTrimFraction	The proportion of loci that are trimmed from the lower end of the range of Fst before the likelihood funciton is applied.
RightTrimFraction	The proportion of loci that are trimmed from the upper end of the range of Fst before the likelihood funciton is applied.
Hmin	The minimum heterozygosity required before including calculations from a locus.
qthreshold	The desired false discovery rate threshold for calculating q-values.
...	additional parameters (see documentation of outflank on github)

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.

Github repository: Whitlock & Lotterhos: https://github.com/whitlock/OutFLANK (Check the readme.pdf within the repository for an explanation. Be aware you now can run OufFLANK from a genlight object)

See Also

util.outflank, util.outflank.plotter, util.outflank.MakeDiploidFSTMat

Examples

```
gl.outflank(bandicoot.gl, plot = TRUE)
```

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, parallel = FALSE, n.cores = 16, v = TRUE)
```

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]
parallel	TRUE if parallel processing is required (does fail under Windows) [default FALSE]
n.cores	Number of cores to use if parallel processing is requested [default 16]
v	– verbose if TRUE, silent if FALSE [default TRUE]

Value

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

Author(s)

Arthur Georges (bugs? Post to https://groups.google.com/d/forum/dartr)

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 (bugs? Post to https://groups.google.com/d/forum/dartr)

Examples

```
library(plotly) #needed for lables="interactive"
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 (bugs? Post to https://groups.google.com/d/forum/dartr)

Examples

```
library(rgl)  #needed for the example
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 (bugs? Post to https://groups.google.com/d/forum/dartr)

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, v = 2)
```

Arguments

gl	– name of the genlight containing the SNP genotypes or genind object containing the presence/absence data [required]
v	– verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log ; 3, progress and results summary; 5, full report [default 2]

Value

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

Author(s)

Arthur Georges (bugs? Post to https://groups.google.com/d/forum/dartr)

Examples

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

`gl.plot` *Plotting genlight object*

Description

This function is based on the glPlot function from adegenet. It simply aims to put labels on the individuals and scales them accordingly (if there are not too many). For arguments please refer to the original adegenet function ?glPlot.

Usage

```
gl.plot(x, indlabels = indNames(x), col = NULL, legend = TRUE,
  posi = "bottomleft", bg = rgb(1, 1, 1, 0.5), ...)
```

Arguments

`x`	– a genlight object
`indlabels`	– labels for individuals. if not provided labels are taken from the first 8 letters from indNames.
`col`	– optional color vector (see ?glPlot)
`legend`	– a logical indicating whether a legend should be added
`posi`	– position of the legend
`bg`	– background color of the legend [default is transparent white]
`...`	— additional arguments passed to glPlot function.

Examples

```
gl.plot(bandicoot.gl[1:30,])
```

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, ind.metafile = NULL, covfilename = NULL,
  nas = "-", topskip = NULL, lastmetric = "RepAvg", probar = TRUE)
```

Arguments

filename	path to file (csv file only currently)
ind.metafile	the name of the file that has entails additional information on individuals. For the required format check
covfilename	depreciated, use ind.metafile parameter
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.
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, ind.metafile = covfilename, probar=TRUE)
}
```

`gl.read.dart.2row` *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.2row(datafile, topskip, nmetavar, nas = "-",
  ind.metafile = NULL, pbar = TRUE, v = 2)
```

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]
`pbar`	– display progress bar [FALSE]
`v`	– verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log ; 3, progress and results summary; 5, full report [default 2]

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 (Post to https://groups.google.com/d/forum/dartr)

Examples

```
## Not run:
gl <- gl.read.dart.2row(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 (bugs? Post to https://groups.google.com/d/forum/dartr)

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.recalc.metrics *Recalculate locus metrics when individuals or populations are deleted from a genlight {adegenet} object*

Description

When individuals are deleted from a genlight object generated by DArT, the locus metrics no longer apply. For example, the Call Rate may be different considering the subset of individuals, compared with the full set. This script recalculates those affected locus metrics, namely, avgPIC, CallRate, freqHets, freqHomRef, freqHomSnp, OneRatioRef, OneRatioSnp, PICRef and PICSnp. Metrics that remain unaltered are RepAvg and TrimmedSeq as they are unaffected by the removal of individuals.

Usage

```
gl.recalc.metrics(x, v = 2)
```

Arguments

x	– name of the genlight object containing SNP genotypes [required]
v	– verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log ; 3, progress and results summary; 5, full report [default 2]

Details

The script optionally removes 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 recalculated locus metadata.

Value

A genlight object with the recalculated locus metadata

Author(s)

Arthur Georges (bugs? Post to https://groups.google.com/d/forum/dartr)

See Also

gl.filter.monomorphs

Examples

```
gl <- gl.recalc.metrics(testset.gl, v=2)
```

---

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 based on information provided in a csv file.

Usage

```
gl.recode.ind(x, ind.recode, recalc = TRUE, mono.rm = TRUE, v = 1)
```

Arguments

x	– 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]
recalc	– Recalculate the locus metadata statistics if any individuals are deleted in the filtering [default TRUE]
mono.rm	– Remove monomorphic loci [default TRUE]
v	– verbosity: 0, silent; 1, brief; 2, verbose [default 1]

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, optionally identifies resultant monomorphic loci or loci with all values missing and deletes them (using gl.filter.monomorphs.r). The script also optionally recalculates statistics made redundant by the deletion of individuals from the dataset.

The script returns a genlight object with the new individual labels and the recalculated locus metadata.

Value

A genlight or genind object with the recoded and reduced data

Author(s)

Arthur Georges (bugs? Post to https://groups.google.com/d/forum/dartr)

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 object {adegenet}*

Description

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

Usage

```
gl.recode.pop(x, pop.recode, recalc = TRUE, mono.rm = TRUE, v = 1)
```

Arguments

`x`	– 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]
`recalc`	– Recalculate the locus metadata statistics if any individuals are deleted in the filtering [default TRUE]
`mono.rm`	– Remove monomorphic loci [default TRUE]
`v`	– verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log ; 3, progress and results summary; 5, full report [default 2]

Details

Individuals are assigned to populations based on the specimen metadata data file (csv) used with gl.read.dart(). 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 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, optionally identifies resultant monomorphic loci or loci with all values missing and deletes them (using gl.filter.monomorphs.r). The script also optionally recalculates the locus metadata as appropriate.

Value

A genlight object with the recoded and reduced data

Author(s)

Arthur Georges (bugs? Post to https://groups.google.com/d/forum/dartr)

See Also

gl.filter.monomorphs

Examples

```
## Not run:
   gl <- gl.recode.pop(gl, 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 (bugs? Post to https://groups.google.com/d/forum/dartr)

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(x, method = "loc", plot = TRUE, v = 2)
```

Arguments

x	– 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"]
plot	specify if a histogram of call rate is to be produced [default TRUE]
v	– verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log ; 3, progress and results summary; 5, full report [default 2]

Value

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

Author(s)

Arthur Georges (Post to https://groups.google.com/d/forum/dartr)

Examples

```
gl.report.callrate(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 (bugs? Post to https://groups.google.com/d/forum/dartr)

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 (bugs? Post to https://groups.google.com/d/forum/dartr)

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 (bugs? Post to https://groups.google.com/d/forum/dartr)

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.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. Also remove loci with have only missing value before running the function.

Usage

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

Arguments

gi	a genlight or genind object created (genlight objects are internally converted via gl2gi to genind)
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. The final output is a data frame that holds all statistics of pairwise LD between loci. (See ?LD in package genetics for details).

Author(s)

Bernd Gruber (bugs? Post to https://groups.google.com/d/forum/dartr)

`gl.report.maf` *Report minor allele frequency (MAF) for each locus in a genlight adegenet object*

Description

Summary of minor allele frequencies across loci is reported as histograms.

Usage

```
gl.report.maf(x, maf.limit = 0.5, ind.limit = 5, loc.limit = 30,
  v = 2)
```

Arguments

`x`	– name of the genlight object containing the SNP data [required]
`maf.limit`	– show histograms maf range <= maf.limit [default 0.5]
`ind.limit`	– show histograms only for populations of size greater than ind.limit [default 5]
`loc.limit`	– show histograms only for populations with more than loc.limit polymorphic loci [default 30]
`v`	– verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log ; 3, progress and results summary; 5, full report [default 2]

Author(s)

Arthur Georges (Post to https://groups.google.com/d/forum/dartr)

Examples

```
f <- gl.report.maf(testset.gl)
```

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 (bugs? Post to https://groups.google.com/d/forum/dartr)

Examples

```
gl2 <- gl.report.monomorphs(testset.gl)
```

gl.report.pa *Report number of private alleles possessed by an individual of unknown provenance*

Description

This script calculates the number of private alleles possessed by a focal individual of unknown provenance when compared to a series of target populations.

Usage

```
gl.report.pa(x, id, nmin = 10, threshold = 0, v = 2)
```

Arguments

x	– name of the input genlight object [required]
id	– identity label of the focal individual whose provenance is unknown [required]
nmin	– minimum sample size for a target population to be included in the analysis [default 10]
threshold	– retain those populations for which the focal individual has private alleles less or equal in number than the threshold [default 0]
v	– verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log ; 3, progress and results summary; 5, full report [default 2]

Details

A private allele is an allele possessed by the focal individual, but absent from the target population. It differs from a fixed allelic difference in that the focal individual may be heterozygous, in which case can share one but not both of its alleles with the target population.

Value

A genlight object containing the focal individual (assigned to population "unknown") and populations for which the focal individual is not distinctive (number of loci with private alleles less than or equal to thresold t.

Author(s)

Arthur Georges (bugs? Post to https://groups.google.com/d/forum/dartr)

Examples

```
# Test run with a focal individual from the Macleay River (EmmacMaclGeor)
x <- gl.report.pa(testset.gl, id="UC_00146", nmin=10, threshold=1, v=3)
```

---

gl.report.pa.pop *Report private alleles (and fixed alleles) per pair of populations*

Description

This function reports seperates the genlight object by populations and reports fixed alleles, the pairwise private alleles and the mean absolute allele frequency differences between pair of population.

Usage

```
gl.report.pa.pop(gl, gl2 = NULL)
```

Arguments

gl	– name of the genlight object containing the SNP data [see Details]
gl2	– if two seperate genlight objects are to be compared this can be provided here [see Details]

Details

if no gl2 is provided, the function uses the pop(gl) hierachy to determine pairs of population, otherwise it runs a single comparison between gl and gl2. Hint: in case you want to run comparison between individuals you can simply redefine your pop(gl) via indNames(gl) [Assuming individual names are unique]

Definition of fixed and private allels

The table shows a cross table of possible cases of allele frequencies between two populations (0=homozygote for Allele 1,x= both Alleles are present, 1=homozygote for Allele 2)

p: cases where there is a private allele in pop1 compared to pop2 (but not vice versa)

f: cases where there is a fixed allele in pop1 (and pop2, as those cases are symmetric)

		pop1		
		0	**x**	**1**
	0	-	p	p,f
pop2	**x**	-	-	-
	1	p,f	p	-

Value

A data.frame will be returned. Each row shows for a pair of populations the number of individuals in a population, the number of loci with fixed differences (same for both populations) in pop1 (compared to pop2) and vice versa. Same for private alleles and finally the absolute mean allele frequendy difference between loci (mdf).

Author(s)

Bernd Gruber (Post to https://groups.google.com/d/forum/dartr)

Examples

```
gl.report.pa.pop(testset.gl[1:20,])
```

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 (bugs? Post to https://groups.google.com/d/forum/dartr)

Examples

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

gl.report.secondaries *Report loci containing secondary SNPs in a genlight {adegenet} object*

Description

SNP datasets generated by DArT include fragments with more than one SNP (that is, with secondaries) 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 secondaries. This script reports duplicate loci.

Usage

```
gl.report.secondaries(gl)
```

Arguments

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

Value

1

Author(s)

Arthur Georges (Post to https://groups.google.com/d/forum/dartr)

Examples

```
gl.report.secondaries(testset.gl)
```

---

gl.sexlinkage — *Identify loci that are sex linked in specimens in a genlight {adegenet} object*

Description

Alleles unique to the Y or W chromosome and monomorphic on the X chromosomes will appear in the SNP dataset as genotypes that are heterozygotic in all individuals of the heterogametic sex and homozygous in all individuals of the homogametic sex.

Usage

```
gl.sexlinkage(x, t.het = 0, t.hom = 0, v = 2)
```

Arguments

x	– name of the genlight object containing the SNP data [required]
t.het	– tolerance, that is tm=0.05 means that 5 be regarded as consistent with a sex specific marker [default 0]
t.hom	– tolerance, that is tf=0.05 means that 5 be regarded as consistent with a sex specific marker [default 0]
v	– verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log ; 3, progress and results summary; 5, full report [default 2]

Details

This script will identify loci with alleles that behave in this way, as putative sex specific SNP markers.

Sex of the individuals for which sex is known with certainty is to be held in the variable x@other$ind.metrics$sex, as M for male, F for female, NA otherwise. The script abbreviates the entries here to the first character. So coding of "Female" and "Male" works as well. Character are also converted to upper cases.

Value

The list of sex specific loci

Author(s)

Arthur Georges (Post to https://groups.google.com/d/forum/dartr)

Examples

```
result <- gl.sexlinkage(testset.gl)
```

`gl.sim.ind` *Simulates individuals based on the allele frequencies provided via a genlight object.*

Description

This function simulates individuals based on the allele frequencies of a genlight object. The output is a genlight object with the same number of loci as the input genlight object.

Usage

```
gl.sim.ind(gl, n = 50, popname = NULL)
```

Arguments

`gl`	– name of the genlight object containing the SNP data
`n`	– number of individuals that should be simulated
`popname`	– a population name for the simulated individuals [default Null]

Details

The function can be used to simulate populations for sampling designs or for power analysis. Check the example below where the effect of drift is explored, by simply simulating several generation a genlight object and putting in the allele frequencies of the previous generation. The beauty of the function is, that it is lightning fast.

Value

a genlight object with n individuals.

Author(s)

Bernd Gruber (bernd.gruber@canberra.edu.au)

Examples

```
glsim <- gl.sim.ind(testset.gl, n=10, popname="sims")
glsim
###Simulate drift over 10 generation
# assuming a bottleneck of only 10 individuals
# [ignoring effect of mating and mutation]
# Simulate 20 individuals with no structure and 50 SNP loci
founder <- glSim(n.ind = 20, n.snp.nonstruc = 50, ploidy=2)
#number of fixed loci in the first generation

res <- sum(colMeans(as.matrix(founder), na.rm=TRUE) %%2 ==0)
simgl <- founder
#49 generations of only 10 individuals
for (i in 2:50)
{
   simgl <- gl.sim.ind(simgl, n=10, popname="sims")
   res[i]<- sum(colMeans(as.matrix(simgl), na.rm=TRUE) %%2 ==0)
}
plot(1:50, res, type="b", xlab="generation", ylab="# fixed loci")
```

---

gl.sim.offspring — *Simulates a specified number of offsprings based on alleles provided by potential father(s) and mother(s)*

Description

This takes a population (or a single individual) of fathers (provided as a genlight object) and mother(s) and simulates offsprings based on "random" mating. It can be used to simulate population dynamics and check the effect of those dynamics and allele frequencies, number of alleles. Another application is to simulate relatedness of siblings and compare it to actual relatedness found in the population to determine kinship.

Usage

```
gl.sim.offspring(fathers, mothers, noffpermother, sexratio = 0.5)
```

Arguments

fathers	– genlight object of potential fathers
mothers	– genlight object of potential mothers simulated
noffpermother	– number of offsprings per mother
sexratio	– the sex ratio of simulated offsprings [females / females +males, 1 equals 100 percent females]

Value

a genlight object with n individuals.

Author(s)

Bernd Gruber (bernd.gruber@canberra.edu.au)

Examples

```
#Simulate 10 potential fathers
gl.fathers <- glSim(10, 20, ploidy=2)
#Simulate 10 potential mothers
gl.mothers <- glSim(10, 20, ploidy=2)
gl.sim.offspring(gl.fathers, gl.mothers, 2, sexratio=0.5)
```

---

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 (bugs? Post to https://groups.google.com/d/forum/dartr)

Examples

```
result <- gl.subsample.loci(testset.gl, n=200, method="avgPIC")
```

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 (bugs? Post to https://groups.google.com/d/forum/dartr)

Examples

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

gl.utils.fdsim *Estimate the rate of false positives in a fixed difference analysis*

Description

This is a support script, called by gl.collapse.recursive. The script takes two populations and generates allele frequency profiles for them. It then samples an allele frequency for each, at random, and estimates a sampling distribution for those two allele frequencies. Drawing two samples from those sampling distributions, it calculates whether or not they represent a fixed difference. This is applied to all loci, and the number of fixed differences so generated are counted, as an expectation. The script distinguished between true fixed differences (with a tolerance of delta), and false positives. The simulation is repeated a given number of times (default=1000) to provide an expectation of the number of false positives, given the observed allele frequency profiles and the sample sizes. The probability of the observed count of fixed differences is greater than the expected number of false positives is calculated.

Usage

```
gl.utils.fdsim(gl, poppair, obs = NULL, reps = 1000, delta = 0.02,
  v = 2)
```

Arguments

gl	– name of the genlight containing the SNP genotypes [required]
poppair	– labels of two populations for comparison in the form c(popA,popB) [required]
obs	– observed number of fixed differences between the two populations [required]
reps	– number of replications to undertake in the simulation [default 1000]
delta	– the threshold value for the minor allele frequency to regard the difference between two populations to be fixed [default 0.02]
v	– verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log ; 3, progress and results summary; 5, full report [default 2]

Value

A list containing the following square matricies [[1]] observed fixed differences; [[2]] mean expected number of false positives for each comparison; [[3]] standard deviation of the no. of false positives for each comparison; [[4]] probability the observed fixed differences arose by chance for each comparison;

Author(s)

Arthur Georges (bugs? Post to https://groups.google.com/d/forum/dartr)

---

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 = "outfile.csv", outpath = tempdir())
```

Arguments

gl	– name of the genlight object containing SNP genotypes [required]
outfile	– name of the csv file to write the data to [required]
outpath	– path where to save the output file (set to tempdir by default)

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 (bugs? Post to https://groups.google.com/d/forum/dartr)

Examples

```
gl.write.csv(testset.gl, outfile="SNP_1row.csv")
```

---

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 (Post to https://groups.google.com/d/forum/dartr)

Examples

```
df <- gl2demerelate(testset.gl)
```

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",
  outpath = tempdir())
```

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]
outpath	– path where to save the output file (set to tempdir by default)

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' to be among the locus metrics (`@other$loc.metrics`) and information of the type of alleles (slot loc.all e.g. "G/A") and the position of the SNP in slot position of the "'genlight"' object (see testset.gl@position and testset.gl@loc.all for how to format these slots.)

Value

A new gl object with all loci rendered homozygous

Author(s)

Bernd Gruber and Arthur Georges (bugs? Post to https://groups.google.com/d/forum/dartr)

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 (Post to https://groups.google.com/d/forum/dartr)

gl2gds *Convert a genlight object to nexus format PAUP SVDquartets*

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", outpath = tempdir())
```

Arguments

gl	– name of the genlight object containing the SNP data [required]
outfile	– file name of the output file (including extension).
outpath	– path where to save the output file (set to tempdir by default)

Author(s)

Arthur Georges (Post to https://groups.google.com/d/forum/dartr)

Examples

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

gl2gi *Converts a genlight object to genind object*

Description

Converts a genlight object to genind object

Usage

```
gl2gi(gl, v = 1)
```

Arguments

gl	– a genlight object
v	– level of verbosity. v=0 is silent, v=1 returns more detailed output during conversion.

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 (bugs? Post to https://groups.google.com/d/forum/dartr)

`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", outpath = tempdir(),
  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]
`outpath`	path where to save the output file (set to tempdir by default)
`bstrap`	Number of bootstrap replicates [default 1]

Value

Matrix of Euclidean distances between populations

Author(s)

Arthur Georges (bugs? Post to https://groups.google.com/d/forum/dartr)

Examples

```
result <- gl2phylip(testset.gl, outfile="test.txt", bstrap=10)
```

gl2plink *Converts a genlight object to plink file format*

Description

This function exports a genlight object into plink format and save it into a file

Usage

```
gl2plink(x, outfile = "plink.csv", outpath = tempdir())
```

Arguments

x	– genlight object
outfile	– name (path) of the output plink file [default plink.csv]
outpath	– path of the output file. Default is to tempdir(). If to be saved in the current working directory change to "."

Author(s)

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

Examples

```
gl2plink(testset.gl)
```

gl2sa *Convert genlight objects to the format used in the SNPassoc package*

Description

This function exports a genlight object into a SNPassoc object. See setupSNP

Usage

```
gl2sa(x)
```

Arguments

x	– genlight object

Value

Returns an object of class 'snp' to be used with **SNPassoc**

Author(s)

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

Examples

```
gl2plink(testset.gl)
```

---

gl2shp *Convert genlight objects to ESRI shapefiles or kml files*

Description

This function exports cordinates in a genlight object to a point shape file (including also individual meta data if available). Coordinates are provided under gl@other$latlong and assumed to be in WGS84 coordinates if no proj4 string is provided.

Usage

```
gl2shp(gl, type = "shp",
  proj4 = "+proj=longlat +ellps=WGS84 +datum=WGS84 +no_defs",
  outfile = "gl", outpath = tempdir(), v = 1)
```

Arguments

gl	– genlight containing lat longs [required]
type	– type of output "kml" or "shp"
proj4	– proj4string of data set. If not provided WGS84 is taken as default. (see spatialreference.org for other projections)
outfile	– name (path) of the output shape file
outpath	– path of the output file. Default is to tempdir(). If to be saved in the current working directory change to "."
v	– verbosity: if v=0 no output, v=1 reports name and path of output file. default 1

Author(s)

Bernd Guber (bugs? Post to https://groups.google.com/d/forum/dartr)

Examples

```
gl2shp(testset.gl)
```

gl2snapp — *Convert a genlight object to nexus format suitable for phylogenetic analysis by SNAPP (via BEAUti)*

Description

The output nexus file contains the snp data and relevant PAUP command lines suitable for BEAUti.

Usage

```
gl2snapp(x, outfile = "snapp.nex", outpath = tempdir(), v = 2)
```

Arguments

x	– name of the genlight object containing the SNP data [required]
outfile	– file name of the output file (including extension).
outpath	– path where to save the output file [default tempdir()]
v	– verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log ; 3, progress and results summary; 5, full report [default 2]

Details

Reference: Bryant, D., Bouckaert, R., Felsenstein, J., Rosenberg, N.A. and RoyChoudhury, A. (2012). Inferring species trees directly from biallelic genetic markers: bypassing gene trees in a full coalescent analysis. Molecular Biology and Evolution 29:1917-1932.

Author(s)

Arthur Georges (Post to https://groups.google.com/d/forum/dartr)

Examples

```
gl2snapp(testset.gl)
```

gl2structure — *Converts genlight objects to STRUCTURE formated files*

Description

This function exports genlight objects to STRUCTURE formatted files (be aware there is a gl2faststruture version as well). It is based on the code provided by Lindsay Clark (see https://github.com/lvclark/R_genetics_conv) and this function is basically a wrapper around her numeric2structure function. See also: Lindsay Clark. (2017, August 22). lvclark/R_genetics_conv: R_genetics_conv 1.1 (Version v1.1). Zenodo: http://doi.org/10.5281/zenodo.846816.

Usage

```
gl2structure(gl, indNames = NULL, addcolumns = NULL, ploidy = 2,
  exportMarkerNames = TRUE, outfile = "gl.str", outpath = tempdir(),
  v = 1)
```

Arguments

gl	– genlight containing lat longs [required]
indNames	– switch if individuals names should be added (defult to indNames in gl)
addcolumns	– additional columns to be added before genotypes
ploidy	– defaults to 2
exportMarkerNames	– switch if loci names should be included (locNames(gl))
outfile	– name (path) of the output shape file
outpath	– path of the output file. Default is to tempdir(). If to be saved in the current working directory change to "."
v	– verbosity: if v=0 no output, v=1 reports name and path of output file. default 1

Author(s)

Bernd Gruber (wrapper) and Lindsay V. Clark [lvclark@illinois.edu]

Examples

```
gl2structure(testset.gl)
```

---

gl2svdquartets *Convert a genlight object to nexus format PAUP SVDquartets*

Description

The output nexus file contains the snp data in one of two forms, depending upon what you regard as most appropriate. One form, that used by Chifman and Kubatko, has two lines per individual, one providing the reference SNP the second providing the alternate SNP (method=1). A second form, recommended by Dave Swofford, has a single line per individual, resolving heterozygous SNPs by replacing them with standard ambiguity codes (method=2).

Usage

```
gl2svdquartets(x, outfile = "svd.nex", outpath = tempdir(),
  method = 2, v = 2)
```

Arguments

x	– name of the genlight object containing the SNP data [required]
outfile	– file name of the output file (including extension).
outpath	– path where to save the output file (set to tempdir by default)
method	– method = 1, nexus file with two lines per individual; method = 2, nexus file with one line per individual, ambiguity codes [default 2]
v	– verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log ; 3, progress and results summary; 5, full report [default 2]

Details

Reference: Chifman, J. and L. Kubatko. 2014. Quartet inference from SNP data under the coalescent, Bioinformatics, 30: 3317-3324

Author(s)

Arthur Georges (Post to https://groups.google.com/d/forum/dartr)

Examples

```
gl2svdquartets(testset.gl[1:20,1:100])
```

---

gl2treemix *Convert a genlight object to a treemix input file*

Description

The output file contains the snp data in the format expected by treemix – see the treemix manual. The file needs to be gzipped before it will be recognised by treemix. Plotting functions can be obtained using source("C:/workspace/R/dartR/R/plotting_funcs.R").

Usage

```
gl2treemix(x, outfile = "treemix_input.gz", outpath = tempdir(),
  v = 2)
```

Arguments

x	– name of the genlight object containing the SNP data [required]
outfile	– file name of the output file (including extension).
outpath	– path where to save the output file (set to tempdir by default)
v	– verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log ; 3, progress and results summary; 5, full report [default 2]

Details

Reference: Pickrell and Pritchard (2012). Inference of population splits and mixtures from genome-wide allele frequency data. PLoS Genetics https://doi.org/10.1371/journal.pgen.1002967

Author(s)

Arthur Georges (Post to https://groups.google.com/d/forum/dartr)

Examples

```
gl2treemix(testset.gl)
```

---

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, tloc = 0)
```

Arguments

s1	– percentage SNP allele frequency for the first population [required]
s2	– percentage SNP allele frequency for the second population [required]
tloc	– threshold value for tolerance 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 (bugs? Post to https://groups.google.com/d/forum/dartr)

See Also

gl.fixed.diff

Examples

```
is.fixed(s1=100, s2=0, tloc=0)
is.fixed(96, 4, tloc=0.05)
```

---

platy *Example data set as text file to be imported into a genlight object*

Description

Check ?read.genetable in pacakge PopGenReport for details on the format.

Format

csv

Author(s)

Bernd Gruber (bugs? Post to https://groups.google.com/d/forum/dartr

Examples

```
library(PopGenReport)
read.csv( paste(.libPaths()[1],"/dartR/extdata/platy.csv",sep="" ))
platy <- read.genetable( paste(.libPaths()[1],"/dartR/extdata/platy.csv",
sep="" ), ind=1, pop=2, lat=3, long=4, other.min=5, other.max=6, oneColPerAll=FALSE,
sep="/")
platy.gl <- (gi2gl(platy))
df.loc <- data.frame(RepAvg = runif(nLoc(platy.gl)), CallRate = 1)
platy.gl@other$loc.metrics <- df.loc
gl.report.repavg(platy.gl)
```

possums.gl — *A simulated genlight object created to run a landscape genetic example*

Description

This a test data set to run a landscape genetics example. It contains 10 populations of 30 individuals each and each individual has 300 loci. There are no covariates for individuals or loci.

Usage

```
possums.gl
```

Format

genlight object

Author(s)

Bernd Gruber (bugs? Post to https://groups.google.com/d/forum/dartr

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 (bugs? Post to https://groups.google.com/d/forum/dartr)

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

Internal function called by gl.read.dart

Usage

```
read.dart(filename, nas = "-", topskip = 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.
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)

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 (bugs? Post to https://groups.google.com/d/forum/dartr

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 (bugs? Post to https://groups.google.com/d/forum/dartr

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 (bugs? Post to https://groups.google.com/d/forum/dartr

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 (bugs? Post to https://groups.google.com/d/forum/dartr

`util.outflank` *OutFLANK: An Fst outlier approach by Mike Whitlock and Katie Lotterhos, University of British Columbia.*

Description

This function is the original implementation of Outflank by Whitlock and Lotterhos. dartR simply provides a convinient wrapper around their functions and an easier install being an r package (for information please refer to their github repository) This method looks for Fst outliers from a list of Fst's for different loci. It assumes that each locus has been genotyped in all populations with approximately equal coverage. OutFLANK estimates the distribution of Fst based on a trimmed sample of Fst's. It assumes that the majority of loci in the center of the distribution are neutral and infers the shape of the distribution of neutral Fst using a trimmed set of loci. Loci with the highest and lowest Fst's are trimmed from the data set before this inference, and the distribution of Fst df/(mean Fst) is assumed to follow a chi-square distribution. Based on this inferred distribution, each locus is given a q-value based on its quantile in the inferred null distribution. The main procedure is called OutFLANK – see comments in that function immediately below for input and output formats. The other functions here are necessary and must be uploaded, but are not necessarily needed by the user directly. Steps:

Usage

```
util.outflank(FstDataFrame, LeftTrimFraction = 0.05,
  RightTrimFraction = 0.05, Hmin = 0.1, NumberOfSamples,
  qthreshold = 0.05)
```

Arguments

`FstDataFrame` A data frame that includes a row for each locus, with columns as follows:

- $LocusName: a character string that uniquely names each locus.
- $FST: Fst calculated for this locus. (Kept here to report the unbased Fst of the results)
- $T1: The numerator of the estimator for Fst (necessary, with $T2, to calculate mean Fst)
- $T2: The denominator of the estimator of Fst
- $FSTNoCorr: Fst calculated for this locus without sample size correction. (Used to find outliers)
- $T1NoCorr: The numerator of the estimator for Fst without sample size correction (necessary, with $T2, to calculate mean Fst)
- $T2NoCorr: The denominator of the estimator of Fst without sample size correction
- $He: The heterozygosity of the locus (used to screen out low heterozygosity loci that have a different distribution)

`LeftTrimFraction`

The proportion of loci that are trimmed from the lower end of the range of Fst before the likelihood funciton is applied.

`RightTrimFraction`
: The proportion of loci that are trimmed from the upper end of the range of Fst before the likelihood funciton is applied.

`Hmin`
: The minimum heterozygosity required before including calculations from a locus.

`NumberOfSamples`
: The number of spatial locations included in the data set.

`qthreshold`
: The desired false discovery rate threshold for calculating q-values.

Value

The function returns a list with seven elements:

- FSTbar: the mean FST inferred from loci not marked as outliers
- FSTNoCorrbar: the mean FST (not corrected for sample size -gives an upwardly biased estimate of FST)
- dfInferred: the inferred number of degrees of freedom for the chi-square distribution of neutral FST
- numberLowFstOutliers: Number of loci flagged as having a signficantly low FST (not reliable)
- numberHighFstOutliers: Number of loci identified as haivng significantly high FST
- results: a data frame with a row for each locus. This data frame includes all the original columns in the data set, and six new ones:
  - $indexOrder (the original order of the input data set),
  - $GoodH (Boolean variable which is TRUE if the expected heterozygosity is greater than the Hemin set by input),
  - $OutlierFlag (TRUE if the method identifies the locus as an outlier, FALSE otherwise), and
  - $q (the q-value for the test of neutrality for the locus)
  - $pvalues (the p-value for the test of neutrality for the locus)
  - $pvaluesRightTail the one-sided (right tail) p-value for a locus

Author(s)

Bernd Gruber (bugs? Post to https://groups.google.com/d/forum/dartr); original implementation of Whitlock & Lotterhos

---

`util.outflank.MakeDiploidFSTMat`
Creates OutFLANK input file from individual genotype info.

Description

Creates OutFLANK input file from individual genotype info.

Usage

```
util.outflank.MakeDiploidFSTMat(SNPmat, locusNames, popNames)
```

Arguments

SNPmat	This is an array of genotypes with a row for each individual. There should be a column for each SNP, with the number of copies of the focal allele (0, 1, or 2) for that individual. If that individual is missing data for that SNP, there should be a 9, instead.
locusNames	A list of names for each SNP locus. There should be the same number of locus names as there are columns in SNPmat.
popNames	A list of population names to give location for each individual. Typically multiple individuals will have the same popName. The list popNames should have the same length as the number of rows in SNPmat.

Value

Returns a data frame in the form needed for the main OutFLANK function.

---

util.outflank.plotter *Plotting functions for Fst distributions after OutFLANK This function takes the output of OutFLANK as input with the OFoutput parameter. It plots a histogram of the FST (by default, the uncorrected FSTs used by OutFLANK) of loci and overlays the inferred null histogram.*

Description

Plotting functions for Fst distributions after OutFLANK

This function takes the output of OutFLANK as input with the OFoutput parameter. It plots a histogram of the FST (by default, the uncorrected FSTs used by OutFLANK) of loci and overlays the inferred null histogram.

Usage

```
util.outflank.plotter(OFoutput, withOutliers = TRUE, NoCorr = TRUE,
  Hmin = 0.1, binwidth = 0.005, Zoom = FALSE,
  RightZoomFraction = 0.05, titletext = NULL)
```

Arguments

OFoutput	The output of the function OutFLANK()
withOutliers	Determines whether the loci marked as outliers (with $OutlierFlag) are included in the histogram.
NoCorr	Plots the distribution of FSTNoCorr when TRUE. Recommended, because this is the data used by OutFLANK to infer the distribution.

Hmin	The minimum heterozygosity required before including a locus in the plot.
binwidth	The width of bins in the histogram.
Zoom	If Zoom is set to TRUE, then the graph will zoom in on the right tail of the distirbution (based on argument RightZoomFraction)
RightZoomFraction	Used when Zoom = TRUE. Defines the proportion of the distribution to plot.
titletext	Allows a test string to be printed as a title on the graph

Value

produces a historgram of the FST

`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 (bugs? Post to https://groups.google.com/d/forum/dartr)

`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 (bugs? Post to https://groups.google.com/d/forum/dartr)

`utils.recalc.avgpic` *A utility script to recalculate the OneRatioRef, OneRatioSnp, PICRef, PICSnp, and AvgPIC by locus after some populations have been deleted.*

Description

The locus metadata supplied by DArT has OneRatioRef, OneRatioSnp, PICRef, PICSnp, and AvgPIC included, but the allelec composition will change when some individuals are removed from the dataset and so the initial statistics will no longer apply. This script recalculates these statistics and places the recalculated values in the appropriate place in the genlight object.

Usage

```
utils.recalc.avgpic(x, v = 2)
```

Arguments

x	– name of the genlight object containing the SNP data [required]
v	– verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log ; 3, progress and results summary; 5, full report [default 2]

Details

If the locus metadata OneRatioRef\Snp, PICRef\Snp and/or AvgPIC do not exist, the script creates and populates them.

Value

The modified genlight object

Author(s)

Arthur Georges (Post to https://groups.google.com/d/forum/dartr)

Examples

```
#result <- dartR:::utils.recalc.avgpic(testset.gl)
```

`utils.recalc.callrate` *A utility script to recalculate the callrate by locus after some populations have been deleted*

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. The locus metadata supplied by DArT has callrate included, but the call rate will change when some individuals are removed from the dataset. This script recalculates the callrate and places these recalculated values in the appropriate place in the genlight object.

Usage

```
utils.recalc.callrate(x, v = 2)
```

Arguments

x	– name of the genlight object containing the SNP data [required]
v	– verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log ; 3, progress and results summary; 5, full report [default 2]

Value

The modified genlight object

Author(s)

Arthur Georges (Post to https://groups.google.com/d/forum/dartr)

Examples

```
#result <- dartR:::utils.recalc.callrate(testset.gl)
```

`utils.recalc.freqhets` *A utility script to recalculate the the frequency of the heterozygous SNPs by locus after some populations have been deleted*

Description

The locus metadata supplied by DArT has FreqHets included, but the frequency of the heterozygotes will change when some individuals are removed from the dataset. This script recalculates the FreqHets and places these recalculated values in the appropriate place in the genlight object. Note that the frequency of the homozygote reference SNPS is calculated from the individuals that could be scored.

Usage

```
utils.recalc.freqhets(x, v = 2)
```

Arguments

x	– name of the genlight object containing the SNP data [required]
v	– verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log ; 3, progress and results summary; 5, full report [default 2]

Value

The modified genlight object

Author(s)

Arthur Georges (Post to https://groups.google.com/d/forum/dartr)

Examples

```
#result <- dartR:::utils.recalc.freqhets(testset.gl)
```

---

utils.recalc.freqhomref

A utility script to recalculate the the frequency of the homozygous reference SNP by locus after some populations have been deleted

---

Description

The locus metadata supplied by DArT has FreqHomRef included, but the frequency of the homozygous reference will change when some individuals are removed from the dataset. This script recalculates the FreqHomRef and places these recalculated values in the appropriate place in the genlight object. Note that the frequency of the homozygote reference SNPS is calculated from the individuals that could be scored.

Usage

```
utils.recalc.freqhomref(x, v = 2)
```

Arguments

x	– name of the genlight object containing the SNP data [required]
v	– verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log ; 3, progress and results summary; 5, full report [default 2]

Value

The modified genlight object

Author(s)

Arthur Georges (Post to https://groups.google.com/d/forum/dartr)

Examples

```
#result <- dartR:::utils.recalc.freqhomref(testset.gl)
```

---

utils.recalc.freqhomsnp

A utility script to recalculate the the frequency of the homozygous alternate SNP by locus after some populations have been deleted

---

Description

The locus metadata supplied by DArT has FreqHomSnp included, but the frequency of the homozygous alternate will change when some individuals are removed from the dataset. This script recalculates the FreqHomSnp and places these recalculated values in the appropriate place in the genlight object. Note that the frequency of the homozygote alternate SNPS is calculated from the individuals that could be scored.

Usage

```
utils.recalc.freqhomsnp(x, v = 2)
```

Arguments

x	– name of the genlight object containing the SNP data [required]
v	– verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log ; 3, progress and results summary; 5, full report [default 2]

Value

The modified genlight object

Author(s)

Arthur Georges (Post to https://groups.google.com/d/forum/dartr)

Examples

```
#result <- dartR:::utils.recalc.freqhomsnp(testset.gl)
```

`utils.recalc.maf` *A utility script to recalculate the the minor allele frequency by locus, typically after some populations have been deleted*

Description

The locus metadata supplied by DArT does not have MAF included, so it is calculated and added to the locus.metadata by this script. The minimum allele frequency will change when some individuals are removed from the dataset. This script recalculates the MAF and places these recalculated values in the appropriate place in the genlight object.

Usage

```
utils.recalc.maf(x, v = 2)
```

Arguments

x	– name of the genlight object containing the SNP data [required]
v	– verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log ; 3, progress and results summary; 5, full report [default 2]

Value

The modified genlight dataset

Author(s)

Arthur Georges (Post to https://groups.google.com/d/forum/dartr)

Examples

```
#f <- dartR:::utils.recalc.maf(testset.gl)
```

Index