Package 'dartR'

April 30, 2021

Type Package

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

Version 1.9.6

Date 2021-04-29

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 (>= 3.5), adegenet (>= 2.0.0), ggplot2

biocViews

Imports stats, methods, utils, plyr, tidyr, MASS, stringr, ape, vegan, SNPRelate, StAMPP, sp, PopGenReport, hierfstat, robustbase, gridExtra, HardyWeinberg, foreach, dplyr, crayon, devtools

Suggests knitr, rmarkdown, rgl, parallel, doParallel, data.table, reshape2, pca3d, dismo, pegas, directlabels, rgdal, leaflet.minicharts, leaflet, rrBLUP, poppr, Rcpp, igraph, qvalue, gdistance, seqinr, pheatmap, RColorBrewer, gplots, vcfR, mmod, plotly, ggthemes, ggrepel, raster

License GPL-2

LazyData true

RoxygenNote 7.1.1

NeedsCompilation no

Author Bernd Gruber [aut, cre],
Arthur Georges [aut],
Jose L. Mijangos [aut],
Peter J. Unmack [ctb],
Oliver Berry [ctb],
Lindsay V. Clark [ctb],
Floriaan Devloo-Delva [ctb]

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

Repository CRAN

Date/Publication 2021-04-30 06:50:08 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

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

Description

Converts a genind object to genlight object

Usage

```
gi2gl(gi, parallel = FALSE, verbose = NULL)
```

Arguments

`gi`	– a genind object
`parallel`	– switch to deactivate parallel version. Default set to FALSE. Might not be worth to run it parallel most of the times.
`verbose`	– verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log ; 3, progress and results summary; 5, full report [default 2]

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 (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)

Examples

```
#for the first 10 loci only
gl.alf(possums.gl[,1:10])
barplot(t(as.matrix(gl.alf(possums.gl[,1:10]))))
```

---

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. Be aware due to a conflict of the amova functions from various packages I had to "hack" StAMPP::stamppAmova to avoid a namespace conflict.

Usage

```
gl.amova(x, distance = NULL, permutations = 100)
```

Arguments

x	– name of the genlight containing the SNP genotypes, with population information [required]
distance	– distance matrix between individuals (if not provided NeisD from StAMPP::stamppNeisD is calculated)
permutations	– 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)

Examples

```
#permuations should be higher, here set to 10 because of speed
gl.amova(bandicoot.gl, permutations=10)
```

---

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,
  unknown,
  nmin = 10,
  dim = NULL,
  alpha = 0.05,
  threshold = 0,
  verbose = 3
)
```

Arguments

x	– name of the input genlight object [required]
unknown	– 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]
verbose	– verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log ; 3, progress and results summary; 5, full report [default 2 or as specified using gl.set.verbosity]

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 (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, unknown="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)

Examples

```
gl.basic.stats(possums.gl)
```

gl.collapse *Collapse a distance matrix by amalgamating populations with pairwise fixed difference count less that a threshold*

Description

This script takes a the file generated by gl.fixed.diff and amalgamates populations with distance less than or equal to a specified threshold. The distance matrix is generated by gl.fixed.diff().

Usage

```
gl.collapse(fd, tpop = 0, tloc = 0, pb = FALSE, verbose = NULL)
```

Arguments

fd	– name of the list of matricies produced by gl.fixed.diff() [required]
tpop	– threshold number of fixed differences above which populatons will not be amalgamated [0]
tloc	– threshold defining a fixed difference (e.g. 0.05 implies 95:5 vs 5:95 is fixed) [0]
pb	– if TRUE, show a progress bar on time consuming loops [FALSE]
verbose	– verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log ; 3, progress and results summary; 5, full report [default 2 or as specified using gl.set.verbosity]

Details

The script then applies the new population assignments 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]] $expfpos – NA's, populated by gl.fixed.diff [by simulation] [[7]] $expfpos – NA's, populated by gl.fixed.diff [by simulation] [[8]] $prob – NA's, populated by gl.fixed.diff [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)
fd
fd2 <- gl.collapse(fd,tpop=1)
fd2
fd3 <- gl.collapse(fd2,tpop=1)
fd3
```

---

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

Description

This script takes the output from gl.collapse 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 demonstrated in the sample set, but non-significant.

Usage

```
gl.collapse.pval(
  fd,
  recode.table = "tmp.csv",
  outpath = tempdir(),
  delta = 0.02,
  reps = 1000,
  alpha = 0.05,
  plot = FALSE,
  verbose = NULL
)
```

Arguments

fd	– name of the list containing the collapsed gl object and associated distance matricies output by gl.collapse run with test=TRUE [required]
recode.table	– name of the new recode.table to receive the new population reassignments arising from the amalgamation of populations [tmp.csv]
outpath	– path where to save the output file [default tempdir(), mandated by CRAN]. Use outpath=getwd() or outpath="." when calling this function to direct output files to your working directory.
delta	– threshold for the level of difference between two populations that will be regarded as operationally fixed [Default 0.02]
reps	number of repetitions in the simulations to estimate false positives. [Default 1000].
alpha	– significance level for test of false positives [default 0.05]
plot	– if TRUE, plot a PCoA with the new groupings [default FALSE]
verbose	– verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log ; 3, progress and results summary; 5, full report [default 2 or as specified using gl.set.verbosity]

Value

A list containing the gl object with the new collapsed populations 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 – the expected count of false positives for each comparison [by simulation], otherwise NAs [[7]] $prob – the significance of the count of fixed differences [by simulation]. These should all be significant (< alpha)

Author(s)

Arthur Georges (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 amalgamates populations with a fixed difference count less than or equal to a specified threshold, tpop. The parameter tpop is used to generate amalgamations based on absolute absence of fixed differences (tpop=0), corroborated fixed differences (tpop=1, recommended), or a higher level of corroboration after examining the distribution of fixed differences across the dataset.

Usage

```
gl.collapse.recursive(x, tloc = 0, tpop = 1, verbose = NULL)
```

Arguments

`x`	– name of the genlight object from which the distance matricies are to be calculated [required]
`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]
`verbose`	– verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log ; 3, progress and results summary; 5, full report [default 2 or as specified using gl.set.verbosity]

Details

The distance matricies are generated by gl.fixed.diff(). The script gl.collapse() examines the distance matrix of fixed differences and amalgamates populations with fixed differences of zero. This is done iteratively until no further amalgamations are possible. If tpop is greater than zero, the script then steps up to tpop=1, that is, examines the distance matrix of fixed differences and amalgamates populations with fixed differences of one or less until no further amalgamations are possible. And so on, until the user specified value of tpop is reached.

A final table is generated with no fixed differences less than or equal to the specified threshold, tpop, and the genlight object is recoded with the new population assignments arising from the amalgamations.

Value

A list containing the final gl object 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]] $expfpos – NA's, populated if required by gl.fixed.diff setting test=T [by simulation]; [[7]] $sdfpos – NA's, populated if required by gl.fixed.diff setting test=T [by simulation]; [[8]] $prob – NA's, populated by gl.fixed.diff setting test=T [by simulation]

Author(s)

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

Examples

```
fd <- gl.collapse.recursive(testset.gl, tloc=0, tpop=2, verbose=3)
```

---

gl.compliance.check *Checks a gl object to see if it complies with dartR expectations, and amends to comply if necessary*

Description

A genlight object used by dartR has a number of requirements that allow functions within the package to operate correctly. The genlight object comprises

Usage

```
gl.compliance.check(x, verbose = NULL)
```

Arguments

x	– name of the input genlight object [required]
verbose	– verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log ; 3, progress and results summary; 5, full report [default 2 or as specified using gl.set.verbosity]

Details

(a) The SNP genotypes or Tag Presence/Absence data (SilicoDArT); (b) An associated dataframe (gl@other$loc.metrics) containing the locus metrics (e.g. Call Rate, Repeatability, etc); (c) An associated dataframe (gl@other$ind.metrics) containing the individual/sample metrics (e.g. sex, latitude, longitude, etc); (d) A specimen identity field (indNames(gl)) with the unique labels applied to each individual/sample; (e) A population assignment (popNames) for each individual/specimen; (f) Flags that indicate whether or not calculable locus metrics have been updated.

This function will check to see that the genlight object conforms to expectation in regard to the above requirements, and if it does not, will rectify it.

Value

A genlight object that conforms to the expectations of dartR

Author(s)

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

Examples

```
x <- gl.compliance.check(testset.gl)
```

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

Examples

```
## Not run:
data(possums.gl)
library(raster)  #needed for that example
landscape.sim <- readRDS(system.file("extdata","landscape.sim.rdata", package="dartR"))
#calculate mean centers of individuals per population
xy <- apply(possums.gl@other$xy, 2, function(x) tapply(x, pop(possums.gl), mean))
cd <- gl.costdistances(landscape.sim, xy, method="leastcost", NN=8)
round(cd,3)

## End(Not run)
```

`gl.define.pop` *Define a new population in a genlight {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, verbose = NULL)
```

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
`verbose`	– verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log ; 3, progress and results summary; 5, full report [default 2 or as specified using gl.set.verbosity]

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 (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.ind *Calculate a distance matrix for individuals defined in an {adegenet} genlight object*

Description

This script calculates various distances between individuals 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) distance.

Usage

```
gl.dist.ind(x, method = NULL, plot = TRUE, verbose = NULL)
```

Arguments

x	– name of the genlight containing the SNP genotypes [required]
method	– Specify distance measure [SNP: Euclidean; P/A: Simple]
plot	– if TRUE, display a histogram and a boxplot of the genetic distances [TRUE]
verbose	– verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log ; 3, progress and results summary; 5, full report [default 2 or as specified using gl.set.verbosity]

Details

The distance measure for SNP data can be one of

Euclidean – Euclidean distance as computed by dist() in stat locus.count – number of loci for which individuals differ, as implemented by dist.gene() in ape allele.count – number of allelic differences between two individuals, as implemented by diss.dist() in poppr relatedness – genetic relatedness between individuals (G matrix), as implemented by A.mat() in rrBLUP

The distance measure for Tag P/A data (binary) can be one of

Simple – simple matching, both 1 or both 0 = 0; one 1 and the other 0 = 1. Presence and absence equally weighted. Jaccard – ignores matching 0, both 1 = 0; one 1 and the other 0 = 1. Absences could be for different reasons. Dice – both 0 = 0; both 1 = 2; one 1 and the other 0 = 1. Absences could be for different reasons. Sometimes called the Czekanowski or Sorensen distance. Phi – binary analogue of the Pearson Correlation coefficient.

Refer to the documentation in the relevant packages listed above.

Value

An object of class 'dist' giving distances between individuals

Author(s)

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

Examples

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

`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) distance.

Usage

```
gl.dist.pop(
  x,
  method = "euclidean",
  plot = TRUE,
  boxplot = "standard",
  range = 1.5,
  binary = FALSE,
  p = NULL,
  verbose = NULL
)
```

Arguments

`x`	– name of the genlight containing the SNP genotypes [required]
`method`	– Specify distance measure [euclidean]
`plot`	– if TRUE, display a histogram of the genetic distances, and a whisker plot [TRUE]
`boxplot`	– if 'standard', plots a standard box and whisker plot; if 'adjusted', plots a boxplot adjusted for skewed distributions ['standard']
`range`	– specifies the range for delimiting outliers [1.5 interquartile ranges]
`binary`	– Perform presence/absence standardization before analysis using decostand [FALSE]
`p`	– The power of the Minkowski distance (typically a value ranging from 0.25 to infinity) [0.5]
`verbose`	– verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log ; 3, progress and results summary; 5, full report [default 2 or as specified using gl.set.verbosity]

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.

Value

An object of class 'dist' giving distances between populations

Author(s)

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

Examples

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

---

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 = FALSE, verbose = NULL)
```

Arguments

x	– name of the genlight object containing SNP genotypes [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 FALSE]
verbose	– verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log ; 3, progress and results summary; 5, full report [default 2 or as specified using gl.set.verbosity]

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

See Also

gl.filter.monomorphs

gl.recalc.metrics

Examples

```
# SNP data
   gl2 <- gl.drop.ind(testset.gl, ind.list=c("AA019073","AA004859"))
# Tag P/A data
  gs2 <- gl.drop.ind(testset.gs, ind.list=c("AA020656","AA19077","AA004859"))
  gs2 <- gl.drop.ind(testset.gs, ind.list=c("AA020656","AA19077","AA004859"),
  mono.rm=TRUE, recalc=TRUE)
```

---

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

Description

The script returns a genlight object with specified loci deleted.

Usage

```
gl.drop.loc(x, loc.list = NULL, first = NULL, last = NULL, verbose = NULL)
```

Arguments

x	– name of the genlight object containing SNP genotypes or presence/absence data [required]
loc.list	– a list of loci to be deleted [required, if loc.range not specified]
first	– first of a range of loci to be deleted [required, if loc.list not specified]
last	– last of a range of loci to be deleted [if not specified, last locus in the dataset]
verbose	– verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log ; 3, progress and results summary; 5, full report [default 2 or as specified using gl.set.verbosity]

Value

A genlight object with the reduced data

Author(s)

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

Examples

```
# SNP data
  gl2 <- gl.drop.loc(testset.gl, loc.list=c("100051468|42-A/T", "100049816-51-A/G"))
# Tag P/A data
  gs2 <- gl.drop.loc(testset.gs, loc.list=c("20134188","19249144"))
```

---

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() or gs.read.dart().

Usage

```
gl.drop.pop(
  x,
  pop.list,
  as.pop = NULL,
  recalc = FALSE,
  mono.rm = FALSE,
  verbose = NULL
)
```

Arguments

x	– name of the genlight object containing SNP genotypes or Tag P/A data (SilicoDArT) [required]
pop.list	– a list of populations to be removed [required]
as.pop	– temporarily assign another metric to represent population for the purposes of deletions [default NULL]
recalc	– Recalculate the locus metadata statistics [default FALSE]
mono.rm	– Remove monomorphic loci [default FALSE]
verbose	– verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log ; 3, progress and results summary; 5, full report [default 2 or as specified using gl.set.verbosity]

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

See Also

gl.filter.monomorphs

gl.recalc.metrics

Examples

```
# SNP data
  gl2 <- gl.drop.pop(testset.gl, pop.list=c("EmsubRopeMata","EmvicVictJasp"))
  gl2 <- gl.drop.pop(testset.gl, pop.list=c("EmsubRopeMata","EmvicVictJasp"),
  mono.rm=TRUE,recalc=TRUE)
  gl2 <- gl.drop.pop(testset.gl, pop.list=c("Male","Unknown"),as.pop="sex")
# Tag P/A data
  gs2 <- gl.drop.pop(testset.gs, pop.list=c("EmsubRopeMata","EmvicVictJasp"))
```

---

gl.edit.recode.ind

Create or edit a individual (=specimen) names, create an recode_ind file amd apply the changes to a genlight object.

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(
  x,
  out.recode.file = NULL,
  outpath = tempdir(),
  recalc = FALSE,
  mono.rm = FALSE,
  verbose = NULL
)
```

Arguments

x	Name of the genlight object for which individuals are to be relabelled.[required]
out.recode.file	Name of the file to output the new individual labels [optional]
outpath	– path where to save the output file [default tempdir(), mandated by CRAN].
recalc	– Recalculate the locus metadata statistics [default TRUE]
mono.rm	– Remove monomorphic loci [default TRUE]
verbose	– verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log ; 3, progress and results summary; 5, full report [default 2 or as specified using gl.set.verbosity]

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.

Use outpath=getwd() or outpath="." when calling this function to direct output files to your working directory.

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, out.recode.file="ind.recode.table.csv")
gl <- gl.edit.recode.ind(testset.gl, out.recode.file="ind.recode.table.csv")

## End(Not run)
```

---

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(
  x,
  pop.recode = NULL,
  out.recode.file = NULL,
  outpath = tempdir(),
  recalc = FALSE,
  mono.rm = FALSE,
  verbose = NULL
)
```

Arguments

x	Name of the genlight object for which populations are to be reassigned.[required]
pop.recode	path to recode file
out.recode.file	Name of the file to output the new individual labels [null]
outpath	– path where to save the output file [default tempdir(), mandated by CRAN].
recalc	– Recalculate the locus metadata statistics if any individuals are deleted [default TRUE]
mono.rm	– Remove monomorphic loci [default TRUE]
verbose	– verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log ; 3, progress and results summary; 5, full report [default 2 or as specified using gl.set.verbosity]

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.

Use outpath=getwd() or outpath="." when calling this function to direct output files to your working directory.

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 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.filter.callrate(
  x,
  method = "loc",
  threshold = 0.95,
  mono.rm = FALSE,
  recalc = FALSE,
  recursive = FALSE,
  plot = TRUE,
  bins = 25,
  verbose = NULL
)
```

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, "pop" to remove loci that fail to meet the specified threshold in any one population [default "loc"]
threshold	– threshold value below which loci will be removed [default 0.95]
mono.rm	– Remove monomorphic loci after analysis is complete [default FALSE]
recalc	– Recalculate the locus metadata statistics if any individuals are deleted in the filtering [default FALSE]
recursive	– Repeatedly filter individuals on call rate, each time removing monomorphic loci. Only applies if method="ind" and mono.rm=TRUE [default FALSE]
plot	specify if histograms of call rate, before and after, are to be produced [default TRUE]
bins	– number of bins to display in histograms [default 25]
verbose	– verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log ; 3, progress and results summary; 5, full report [default 2, unless specified using gl.set.verbosity]

Details

Tag Presence/Absence datasets (SilicoDArT) have missing values where it is not possible to determine reliably if there the sequence tag can be called at a particular locus.

method = 'ind': Because this filter operates on call rate, this function recalculates Call Rate, if necessary, before filtering. If individuals are removed using method='ind', then the call rate stored in the genlight object is, optionally, recalcuated after filtering.

recursive=TRUE: 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. This is done by setting mono.rm=TRUE and recursive=TRUE, or it can be done manually.

Value

The reduced genlight or genind object, plus a summary

Author(s)

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

Examples

```
# SNP data
  result <- gl.filter.callrate(testset.gl, method="loc", threshold=0.95, verbose=3)
  result <- gl.filter.callrate(testset.gl, method="ind", threshold=0.8, verbose=3)
# Tag P/A data
  result <- gl.filter.callrate(testset.gs, method="loc", threshold=0.95, verbose=3)
  result <- gl.filter.callrate(testset.gs, method="ind", threshold=0.8, verbose=3)
```

---

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 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, threshold = 0.2, rs = 5, pb = FALSE, verbose = NULL)
```

Arguments

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

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

```
# SNP data
  result <- gl.filter.hamming(testset.gl, threshold=0.25, verbose=3)
```

---

gl.filter.heterozygosity

Filters individuals with average heterozygosity greater than a specified upper threshold or less than a specified lower threshold.

Description

Calculates the observed heterozyosity for each individual in a genlight object and filters individuals based on specified threshold values. Use gl.report.heterozygosity to determine the appropriate thresholds.

Usage

```
gl.filter.heterozygosity(x, t.upper = 0.7, t.lower = 0, verbose = NULL)
```

Arguments

x	– a genlight object containing the SNP genotypes [Required]
t.upper	– filter individuals > the threshold [default 0.7]
t.lower	– filter individuals < the threshold [default 0]
verbose	– verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log ; 3, progress and results summary; 5, full report [default 2, unless specified using gl.set.verbosity]

Value

the filtered genlight object

Author(s)

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

Examples

```
result <- gl.filter.heterozygosity(testset.gl,t.upper=0.06,verbose=3)
tmp <- gl.report.heterozygosity(result,method="ind")
```

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, verbose = NULL)
```

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]
verbose	– verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log ; 3, progress and results summary; 5, full report [default 2, unless specified using gl.set.verbosity]

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

```
result <- gl.filter.hwe(testset.gl, 0.05, bon=TRUE, verbose=3)
```

`gl.filter.locmetric` *Filter loci on the basis of numeric information stored in other$loc.metrics in a genlight {adegenet} object*

Description

This script uses any field with numeric values stored in $other$loc.metrics to filter loci. The loci to keep can be within the upper and lower thresholds ("within") or outside of the upper and lower thresholds ("outside"). The fields that are included in dartR, and a short description, are found below. Optionally, the user can also set his/her own filter by adding a vector into $other$loc.metrics as shown in the example.

Usage

```
gl.filter.locmetric(x, metric, upper, lower, keep = "within", verbose = 2)
```

Arguments

`x`	– name of the genlight object containing the SNP data [required]
`metric`	– name of the metric to be used for filtering [required]
`upper`	– filter upper threshold [required]
`lower`	– filter lower threshold [required]
`keep`	– whether keep loci within of upper and lower thresholds or keep loci outside of upper and lower thresholds [within]
`verbose`	– verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log ; 3, progress and results summary; 5, full report [default 2, unless specified using gl.set.verbosity]

Details

- SnpPosition - position (zero is position 1) in the sequence tag of the defined SNP variant base - CallRate - proportion of samples for which the genotype call is non-missing (that is, not '-') - OneRatioRef - proportion of samples for which the genotype score is 0 - OneRatioSnp - proportion of samples for which the genotype score is 2 - FreqHomRef - proportion of samples homozygous for the Reference allele - FreqHomSnp - proportion of samples homozygous for the Alternate (SNP) allele - FreqHets - proportion of samples which score as heterozygous, that is, scored as 1 - PICRef - polymorphism information content (PIC) for the Reference allele - PICSnp - polymorphism information content (PIC) for the SNP - AvgPIC - average of the polymorphism information content (PIC) of the Reference and SNP alleles - AvgCountRef - sum of the tag read counts for all samples, divided by the number of samples with non-zero tag read counts, for the Reference allele row - AvgCountSnp - sum of the tag read counts for all samples, divided by the number of samples with non-zero tag read counts, for the Alternate (SNP) allele row - RepAvg - proportion of technical replicate assay pairs for which the marker score is consistent

Value

The reduced genlight dataset

Author(s)

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

Examples

```
# adding dummy data
test <- testset.gl
test$other$loc.metrics$test <- 1:nLoc(test)
result <- gl.filter.locmetric(x=test, metric= "test", upper=255,
lower=200, keep= "within", verbose=3)
```

---

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, verbose = NULL)
```

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]
verbose	– verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log ; 3, progress and results summary; 5, full report [default 2, unless specified using gl.set.verbosity]

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

```
result <- gl.filter.monomorphs(testset.gl)
result <- gl.filter.maf(result, threshold=0.05, verbose=3)
```

---

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, verbose = NULL)
```

Arguments

x	– name of the input genlight object [required]
verbose	– verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log ; 3, progress and results summary; 5, full report [default 2, unless specified using gl.set.verbosity]

Details

A DArT dataset will not have monomorphic loci, but they can arise, along with loci that are scored all NA, when populations or individuals are deleted. Retaining monomorphic loci unnecessarily increases the size of the dataset and will affect some calculations.

Note that for SNP data, NAs likely represent null alleles; in tag presence/absence data, NAs represent missing values (presence/absence could not be reliably scored)

Value

A genlight object with monomorphic (and all NA) loci removed

Author(s)

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

Examples

```
# SNP data
  result <- gl.filter.monomorphs(testset.gl, verbose=3)
# Tag P/A data
  result <- gl.filter.monomorphs(testset.gs, verbose=3)
```

`gl.filter.overshoot` *Filters loci for which the SNP has been trimmed from the sequence tag along with the adaptor*

Description

This function checks the position of the SNP within the trimmed sequence tag and identifies those for which the SNP position is outside the trimmed sequence tag. This can happen, rarely, when the sequence containing the SNP resembles the adaptor.

Usage

```
gl.filter.overshoot(x, verbose = NULL)
```

Arguments

x	– name of the genlight object [required]
verbose	– verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log ; 3, progress and results summary; 5, full report [default 2, unless specified using gl.set.verbosity]

Details

The SNP genotype can still be used in most analyses, but functions like gl2fasta() will present challenges if the SNP has been trimmed from the sequence tag.

Not fatal, but should apply this filter before gl.filter.secondaries, for obvious reasons.

Value

A new genlight object with the recalcitrant loci deleted

Author(s)

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

Examples

```
result <- gl.filter.overshoot(testset.gl, verbose=3)
```

`gl.filter.pa` *Filter loci that contain private (and fixed alleles) between two populations.*

Description

This script is meant to be used prior to `gl.nhybrids` to maximise the information content of the snps used to identify hybrids (currently newhybrids does allow only 200 SNPs). The idea is to use first all loci that have fixed alleles between the potential source populations and then "fill up" to 200 loci using loci that have private alleles between those. The functions filters for those loci (if invers is set to TRUE, the opposite is returned (all loci that are not fixed and have no private alleles - not sure why yet, but maybe useful.)

Usage

```
gl.filter.pa(x, pop1, pop2, invers = FALSE, verbose = NULL)
```

Arguments

`x`	– name of the genlight object containing the SNP data [required]
`pop1`	– name of the first parental population (in quotes) [required]
`pop2`	– name of the second parental population (in quotes) [required]
`invers`	– switch to filter for all loci that have no private alleles and are not fixed [FALSE]
`verbose`	– verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log ; 3, progress and results summary; 5, full report [default 2, unless specified using gl.set.verbosity]

Value

The reduced genlight dataset, containing now only fixed and private alleles

Author(s)

Bernd Gruber & Ella Kelly (University of Melbourne) (Post to https://groups.google.com/d/forum/dartr)

Examples

```
result <- gl.filter.pa(testset.gl, pop1=pop(testset.gl)[1], pop2=pop(testset.gl)[2],verbose=3)
```

gl.filter.parent.offspring

Filter putative parent offspring within a population

Description

This script removes individuals suspected of being related as parent-offspring. It examines the frequency of pedigree inconsistent loci, that is, those loci that are homozygotes in the parent for the reference allele, and homozygous in the offspring for the alternate allele. This condition is not consistent with any pedigree, regardless of the (unkonwn) genotype of the other parent. The pedigree inconsistent loci are counted as an indication of whether or not it is reasonable to propose the two individuals are in a parent-offspring relationship.

Usage

```
gl.filter.parent.offspring(
  x,
  min.rdepth = 12,
  min.reproducibility = 1,
  range = 1.5,
  rm.monomorphs = FALSE,
  verbose = NULL
)
```

Arguments

x	Name of the genlight object containing the SNP genotypes [required]
min.rdepth	Minimum read depth to include in analysis [default = 12]
min.reproducibility	Minimum reproducibility to include in analysis [default = 1]
range	– specifies the range to extend beyond the interquartile range for delimiting outliers [default = 1.5 interquartile ranges]
rm.monomorphs	– if TRUE, remove monomorphic loci after filtering individuals [default FALSE]
verbose	– verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log ; 3, progress and results summary; 5, full report [default 2 or as specified using gl.set.verbosity]

Details

Obviously, if the two individuals are in a parent offspring relationship, the true number of pedigree inconsistent loci should be zero, but SNP calling is not infallible. Some loci will be mis-called. The problem thus becomes one of determining if the two focal individuals have a count of pedigree inconsistent loci less than would be expected of typical unrelated individuals. There are some quite sophisticated software packages available to formally apply likelihoods to the decision, but we use a simple outlier comparison.

To reduce the frequency of mis-calls, and so emphasise the difference between true parent-offspring pairs and unrelated pairs, the data can be filtered on read depth. Typically minimum read depth is set to 5x, but you can examine the distribution of read depths with gl.report.rdepth() and push this up with an acceptable loss of loci. 12x might be a good minimum for this particular analysis. It is sensible also to push the minimum reproducibility up to 1, if that does not result in an unacceptable loss of loci.

Note that the null expectation is not well defined, and the power reduced, if the population from which the putative parent-offspring pairs are drawn contains many sibs. Note also that if an individual has been genotyped twice in the dataset, the replicate pair will be assessed by this script as being in a parent-offspring relationship.

You should run gl.report.parent.offspring() before filtering. Use this report to decide min.rdepth and min.reproducibility and assess impact on your dataset.

Value

A set of individuals in parent-offspring relationship

---

gl.filter.rdepth *Filter loci based on counts of sequence tags scored at a locus (read depth)*

Description

SNP datasets generated by DArT report AvgCountRef and AvgCountSnp as counts of sequence tags for the reference and alternate alleles respectively. These can be used to backcalculate Read Depth. Fragment presence/absence datasets as provided by DArT (SilicoDArT) provide Average Read Depth and Standard Deviation of Read Depth as stanard columns in their report.

Usage

```
gl.filter.rdepth(x, lower = 5, upper = 50, verbose = NULL)
```

Arguments

x	– name of the genlight object containing the SNP or tag presence/absence data [required]
lower	– lower threshold value below which loci will be removed [default 5]
upper	– upper threshold value above which loci will be removed [default 50]
verbose	– verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log ; 3, progress and results summary; 5, full report [default 2, unless specified using gl.set.verbosity]

Details

Filtering on Read Depth using the companion script gl.filter.rdepth can be on the basis of loci with exceptionally low counts, or loci with exceptionally high counts.

Value

Returns a genlight object retaining loci with a Read Depth in the range specified by the lower and upper threshold.

Author(s)

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

Examples

```
# SNP data
  gl.report.rdepth(testset.gl)
  result <- gl.filter.rdepth(testset.gl, lower=8, upper=50, verbose=3)
# Tag P/A data
  result <- gl.filter.rdepth(testset.gs, lower=8, upper=50)
```

---

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 an index, RepAvg, generated by reproducing the data independently for 30 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, verbose = NULL)
```

Arguments

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

Details

SilicoDArT datasets generated by DArT have a smilar index, Reproducibility. For these fragment presence/absence data, repeatability is the percentage of scores that are repeated in the technical replicate dataset.

Value

Returns a genlight object retaining loci with repeatability (Repavg or Reproducibility) greater than the specified threshold.

Author(s)

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

Examples

```
# SNP data
  gl.report.reproducibility(testset.gl)
  result <- gl.filter.reproducibility(testset.gl, threshold=0.99, verbose=3)
# Tag P/A data
  gl.report.reproducibility(testset.gs)
  result <- gl.filter.reproducibility(testset.gs, threshold=0.99)
```

---

gl.filter.reproducibility

Filter loci in a genlight {adegenet} object based on average repeatability of alleles at a locus

---

Description

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

Usage

```
gl.filter.reproducibility(x, threshold = 0.99, verbose = NULL)
```

Arguments

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

Details

SilicoDArT datasets generated by DArT have a smilar index, Reproducibility. For these fragment presence/absence data, repeatability is the percentage of scores that are repeated in the technical replicate dataset.

Value

Returns a genlight object retaining loci with repeatability (Repavg or Reproducibility) greater than the specified threshold.

Author(s)

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

Examples

```
# SNP data
  gl.report.reproducibility(testset.gl)
  result <- gl.filter.reproducibility(testset.gl, threshold=0.99, verbose=3)
# Tag P/A data
  gl.report.reproducibility(testset.gs)
  result <- gl.filter.reproducibility(testset.gs, threshold=0.99)
```

---

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.

Usage

```
gl.filter.secondaries(x, method = "random", verbose = NULL)
```

Arguments

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

Details

This script filters out all but the first sequence tag with the same CloneID after ordering the genlight object on based on repeatability, avgPIC in that order (method="best") or at random (method="random").

The filter has not been implemented for tag presence/absence data.

Value

The reduced genlight, plus a summary

Author(s)

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

Examples

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

`gl.filter.sexlinked` *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.filter.sexlinked(
  x,
  sex = NULL,
  filter = NULL,
  read.depth = 0,
  t.het = 0,
  t.hom = 0,
  t.pres = 0,
  plot = FALSE,
  verbose = NULL
)
```

Arguments

`x`	– name of the genlight object containing the SNP data [required]
`sex`	– factor that defines the sex of individuals. See explanation above.
`filter`	– switch to either 'keep' sexlinked markers only from the genlight object or drop them. Default is NULL, so has to be specified.
`read.depth`	– additional filter option to keep only loci above a certain read.depth. Default to 0, which means read.depth is not taken into account.
`t.het`	– tolerance, that is t.het=0.05 means that 5% of the heterogametic sex can be homozygous and still be regarded as consistent with a sex specific marker [default 0]
`t.hom`	– tolerance, that is t.hom=0.05 means that 5% of the homogametic sex can be heterozygous and still be regarded as consistent with a sex specific marker [default 0]
`t.pres`	– tolerance, that is t.pres=0.05 means that a silicodart marker can be present in either of the sexes and still be regarded as a sex-linked marker. [default 0]

plot	– creates a plot that shows the heterozygosity of males and females at each loci. be regarded as consistent with a sex specific marker [default 0]
verbose	– 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 can be provided via a factor (equal to the length of the number of individuals) or to be held in the variable `x@other$ind.metrics$sex`. Coding is: M for male, F for female, U or NA for unknown/missing. 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 filtered genlight object (filter="keep": sexlinked loci,filter="drop", everything except sexlinked loci).

Author(s)

Arthur Georges, Bernd Gruber & Floriaan Devloo-Delvan (Post to https://groups.google.com/d/forum/dartr)

Examples

```
cat("does not work yet")
#result <- gl.sexlinkage(testset.gl)
```

---

gl.filter.taglength *Filter loci in a genlight {adegenet} object based on sequence tag length*

Description

SNP datasets generated by DArT typically have sequence tag lengths ranging from 20 to 69 base pairs.

Usage

```
gl.filter.taglength(x, lower = 20, upper = 69, verbose = NULL)
```

Arguments

x	– name of the genlight object containing the SNP data [required]
lower	– lower threshold value below which loci will be removed [default 20]
upper	– upper threshold value above which loci will be removed [default 69]
verbose	– verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log ; 3, progress and results summary; 5, full report [default 2, unless specified using gl.set.verbosity]

Value

Returns a genlight object retaining loci with a sequence tag length in the range specified by the lower and upper threshold.

Author(s)

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

Examples

```
# SNP data
  gl.report.taglength(testset.gl)
  result <- gl.filter.taglength(testset.gl,lower=60)
  gl.report.taglength(result)
# Tag P/A data
  gl.report.taglength(testset.gs)
  result <- gl.filter.taglength(testset.gs,lower=60)
  gl.report.taglength(result)
```

---

gl.fixed.diff *Generate a matrix of fixed differences*

Description

This script takes SNP data or sequence tag P/A 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,
  alpha = 0.05,
  reps = 1000,
  mono.rm = TRUE,
  pb = TRUE,
  verbose = NULL
)
```

Arguments

x	– name of the genlight object containing SNP genotypes or tag P/A data (SilicoDArT) or an object of class 'fd' [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	– threshold value for the true population minor allele frequency (MAF) from which resultant sample fixed differences are considered true positives [default 0.02]
alpha	– level of significance used to display non-significant differences between populations as they are compared pairwise [default 0.05]
reps	– number of replications to undertake in the simulation to estimate probability of false positives [default 1000]
mono.rm	– if TRUE, loci that are monomorphic across all individuals are removed before beginning computations [default TRUE]
pb	– if TRUE, show a progress bar on time consuming loops [default FALSE]
verbose	– verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log ; 3, progress and results summary; 5, full report [default 2 or as specified using gl.set.verbosity]

Details

A fixed difference at a locus occurs when two populations share no alleles or where all members of one population has a sequence tag scored, and all members of the other population has the sequence tag absent. 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 are 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.

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).

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 of Class "fd" containing the gl object and square matricies, as follows [[1]] $gl – the output 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]] $expfpos – if test=TRUE, the expected count of false positives for each comparison [by simulation]; [[7]] $sdfpos – if test=TRUE, the standard deviation of the count of false positives for each comparison [by simulation]; [[8]] $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, verbose=2 )
fd <- gl.fixed.diff(testset.gl, tloc=0, test=TRUE, delta=0.02, reps=100, verbose=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 <- 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)
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 an identity by descent matrix*

Description

This function calculates the mean probability of identity by descent (IBD) across loci that would result from all the possible crosses of the individuals analyzed. IBD is calculated by an additive relationship matrix approach developed by Endelman and Jannink (2012) as implemented in the function A.mat (package rrBLUP). Two or more alleles are identical by descent (IBD) if they are identical copies of the same ancestral allele in a base population. The additive relationship matrix is a theoretical framework for estimating a relationship matrix that is consistent with an approach to estimate the probability that the alleles at a random locus are identical in state (IBS). This function also plots a heatmap, and a dendrogram, of IBD values where each diagonal element has a mean that equals 1+f, where f is the inbreeding coefficient (i.e. the probability that the two alleles at a randomly chosen locus are IBD from the base population). As this probability lies between 0 and 1, the diagonal elements range from 1 to 2. Because the inbreeding coefficients are expressed relative to the current population, the mean of the off-diagonal elements is -(1+f)/n, where n is the number of loci. Individual names are shown in the margins of the heatmap and colors represent different populations.

Usage

```
gl.grm(x, plotheatmap = TRUE, verbose = NULL, ...)
```

Arguments

`x`	– a genlight object
`plotheatmap`	– a switch if a heatmap should be shown [Default:TRUE]
`verbose`	– verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log ; 3, progress and results summary; 5, full report [default 2 or as specified using gl.set.verbosity]
`...`	parameters passed to function A.mat from package rrBLUP

Value

an identity by descent matrix and a heatmap plot

References

Endelman, J. B. (2011). Ridge regression and other kernels for genomic selection with r package rrblup. The Plant Genome 4, 250. Endelman, J. B. , Jannink, J.-L. (2012). Shrinkage estimation of the realized relationship matrix. G3: Genes, Genomics, Genetics 2, 1405.

Examples

```
gl.grm(bandicoot.gl[1:20,])
```

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",
  verbose = 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"]
verbose	– 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 (Post to https://groups.google.com/d/forum/dartr)

Examples

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

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

Description

A very simple function to report expected Heterozygosity

Usage

```
gl.He(gl)
```

Arguments

`gl`	– a genlight object

Value

a simple vector wiht Ho for each loci

Author(s)

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

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.hwe.pop — *Function to run Hardy-Weinberg tests over each loci and population*

Description

Does a Hardy-Weinberg-Test for each loci in each of the populations as defined by the pop slot in a genlight object. The function is completely rewritten as the previous function was based on package SNPassoc, which no longer is supported by CRAN. It now employs the HardyWeinberg package, which needs to be installed. The function that is used is HWExactStats, but there are several other great function implemented in the package regarding HWE. Therefore you can return the data in the format the HWE package expects, via HWformat=TRUE and then use this to run other functions of the package.

Usage

```
gl.hwe.pop(x, pvalue = 0.05, plot = TRUE, HWformat = FALSE)
```

Arguments

x	a genlight object with a population defined [pop(x) does not return NULL]
pvalue	the p-value for the HWE test.
plot	a switch if a barplot is returned.
HWformat	switch if data should be returned in HWformat (counts of Genotypes to be used in package HardyWeinberg)

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. The function returns a list with up to three components: 'HWE' is the matrix over loci and populations, 'plot' is a matrixplot (ggplot) which shows the significant results for population and loci (can be amended further using ggplot syntax), and finally if 'HWEformat=TRUE' the 'HWformat' entails SNP data for each population in 'HardyWeinberg'-format to be used with other functions of the package (e.g HWPerm or HWExactPrevious).

Examples

```
out <- gl.hwe.pop(bandicoot.gl[,1:33], pvalue=0.05, plot=TRUE, HWformat=FALSE)
```

---

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(
  x,
  Dgen = NULL,
  Dgeo = NULL,
  projected = FALSE,
  permutations = 999,
  plot = TRUE
)
```

Arguments

x	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.install.vanilla.dartR

This functions installs all required packages for using all functions available in dartR The function compares the installed packages with the the currently available ones on cran. Be aware this function only works if a version of dartR is already installed on your system. You can choose if you also want to have a specific version of dartR installed ("CRAN", "master" or "dev"). "master" and "dev" are installed from Github. Be aware the dev version from github is not fully tested and most certainly will contain untested functions.

Description

This functions installs all required packages for using all functions available in dartR

The function compares the installed packages with the the currently available ones on cran. Be aware this function only works if a version of dartR is already installed on your system. You can choose if you also want to have a specific version of dartR installed ("CRAN", "master" or "dev"). "master" and "dev" are installed from Github. Be aware the dev version from github is not fully tested and most certainly will contain untested functions.

Usage

```
gl.install.vanilla.dartR(flavour = NULL, verbose = TRUE)
```

Arguments

flavour	– If and which version of R you want to install. If NULL then only needed packages for the current version will be installed. If "CRAN" current CRAN version will be installed. "master" installs the GitHub master branch and "dev" installs the experimental development branch from GitHub.
verbose	returns information on packages and dartR versions

Value

returns a message if the installation was successful/required

Author(s)

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

---

gl.join *Combines two genlight objects*

Description

This function combines two genlight objects and their associated metadata. The history associated with the two genlight objects is cleared from the new genlight object. The individuals/samples must be the same in each genlight object.

Usage

```
gl.join(x1, x2, verbose = NULL)
```

Arguments

x1	– name of the first genlight object [required]
x2	– name of the first genlight object [required]
verbose	– verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log ; 3, progress and results summary; 5, full report [default 2 or as specified using gl.set.verbosity]

Details

The function is typically used to combine datasets from the same service where the files have been split because of size limitations. The data is read in from multiple csv files, then the resultant genlight objects are combined.

Value

A new genlight object

Author(s)

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

Examples

```
x1 <- testset.gl[,1:100]
x1@other$loc.metrics <-  testset.gl@other$loc.metrics[1:100,]
nLoc(x1)
x2 <- testset.gl[,101:150]
x2@other$loc.metrics <-  testset.gl@other$loc.metrics[101:150,]
nLoc(x2)
gl <- gl.join(x1, x2, verbose=2)
nLoc(gl)
```

---

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 = FALSE, verbose = NULL)
```

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 FALSE]
verbose	– verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log ; 3, progress and results summary; 5, full report [default 2 or as specified using gl.set.verbosity]

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

See Also

gl.filter.monomorphs for when mono.rm=TRUE, gl.recalc.metrics for when recalc=TRUE

gl.drop.ind to drop rather than keep specified individuals

Examples

```
# SNP data
  gl2 <- gl.keep.ind(testset.gl, ind.list=c("AA019073","AA004859"))
# Tag P/A data
 gs2 <- gl.keep.ind(testset.gs, ind.list=c("AA020656","AA19077","AA004859"))
```

---

gl.keep.loc *Remove all but the specified loci from a genelight {adegenet} object*

Description

The script returns a genlight object with the all but the specified loci deleted.

Usage

```
gl.keep.loc(x, loc.list = NULL, first = NULL, last = NULL, verbose = NULL)
```

Arguments

x	– name of the genlight object containing SNP genotypes or presence/absence data [required]
loc.list	– a list of loci to be kept [required, if loc.range not specified]
first	– first of a range of loci to be kept [required, if loc.list not specified]
last	– last of a range of loci to be kept [if not specified, last locus in the dataset]
verbose	– verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log ; 3, progress and results summary; 5, full report [default 2 or as specified using gl.set.verbosity]

Value

A genlight object with the reduced data

Author(s)

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

Examples

```
# SNP data
  gl2 <- gl.keep.loc(testset.gl, loc.list=c("100051468|42-A/T", "100049816-51-A/G"))
# Tag P/A data
  gs2 <- gl.keep.loc(testset.gs, loc.list=c("20134188","19249144"))
```

---

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,
  as.pop = NULL,
  recalc = FALSE,
  mono.rm = FALSE,
  verbose = NULL
)
```

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]
as.pop	– assign another metric to represent population [default NULL]
recalc	– Recalculate the locus metadata statistics [default FALSE]
mono.rm	– Remove monomorphic loci [default FALSE]
verbose	– verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log ; 3, progress and results summary; 5, full report [default 2 or as specified using gl.set.verbosity]

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

See Also

gl.filter.monomorphs for when mono.rm=TRUE, gl.recalc.metrics for when recalc=TRUE

gl.drop.pop to drop rather than keep specified populations

Examples

```
# SNP data
  gl2 <- gl.keep.pop(testset.gl, pop.list=c("EmsubRopeMata","EmvicVictJasp"))
  gl2 <- gl.keep.pop(testset.gl, pop.list=c("EmsubRopeMata","EmvicVictJasp"),
  mono.rm=TRUE,recalc=TRUE)
  gl2 <- gl.keep.pop(testset.gl, pop.list=c("Female"),as.pop="sex")
# Tag P/A data
  gs2 <- gl.keep.pop(testset.gs, pop.list=c("EmsubRopeMata","EmvicVictJasp"))
```

---

gl.load — *Retrieves an object in compressed binary format earlier saved using gl.save.*

Description

This is a wrapper for readRDS().

Usage

```
gl.load(file, verbose = NULL)
```

Arguments

file	– name of the binary file from which to retrieve the object [required]
verbose	– verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log ; 3, progress and results summary; 5, full report [default 2 or as specified using gl.set.verbosity]

Details

The script retrieves the object from the current workspace and returns it by assignment.

Value

the retrieved object

Author(s)

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

Examples

```
gl.save(testset.gl,file.path(tempdir(),"testset.rds"))
gl.reloaded <- gl.load(file.path(tempdir(),"testset.rds"))
```

---

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,
  out.recode.file = "default_recode_ind.csv",
  outpath = tempdir(),
  verbose = NULL
)
```

Arguments

x	– name of the genlight object containing the SNP data [required]
out.recode.file	– file name of the output file (including extension) [default default_recode_ind.csv]
outpath	– path where to save the output file [default tempdir(), mandated by CRAN]. Use outpath=getwd() or outpath="." when calling this function to direct output files to your working directory.
verbose	– verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log ; 3, progress and results summary; 5, full report [default 2 or as specified using gl.set.verbosity]

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, out.recode.file ="Emmac_recode_ind.csv",outpath=tempdir())
```

---

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,
  out.recode.file = "recode_pop_table.csv",
  outpath = tempdir(),
  verbose = NULL
)
```

Arguments

x	– name of the genlight object containing the SNP data [required]
out.recode.file	– file name of the output file (including extension) [default recode_pop_table.csv]
outpath	– path where to save the output file [default tempdir(), mandated by CRAN]. Use outpath=getwd() or outpath="." when calling this function to direct output files to your working directory.

verbose – verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log ; 3, progress and results summary; 5, full report [default 2 or as specified using gl.set.verbosity]

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,out.recode.file="test.csv",outpath=tempdir(),verbose=2)
```

---

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,
  matrix = NULL,
  standard = TRUE,
  symmetric = TRUE,
  ind.circles = TRUE,
  pop.labels = TRUE,
  pop.labels.cex = 12,
  provider = "Esri.NatGeoWorldMap"
)
```

Arguments

x	– a genlight object [including coordinates within the latlong slot]
matrix	– a distance matrix between populations or individuals. The matrix is visualised as lines between individuals/populations. If matrix is asymmetric two lines with arrows are plotted.
standard	– if a matrix is provided line width will be standardised to be between 1 to 10, if set to true, otherwise taken as given.
symmetric	– if a symmetric matrix is provided only one line is drawn based on the lower triangle of the matrix. If set to false arrows indicating the direction are used instead.
ind.circles	– should individuals plotted as circles, default is TRUE
pop.labels	– population labels at the center of the individuals of populations, default is TRUE
pop.labels.cex	– size of population labels, default is 20.
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.interactive(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, verbose = NULL)
```

Arguments

x	– name of the genlight object containing SNP genotypes [required]
old	– a list of populations to be merged [required]
new	– name of the new population [required]
verbose	– verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log ; 3, progress and results summary; 5, full report [default 2 or as specified using gl.set.verbosity]

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 (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 (or hard specified loc.limit) 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 (method="AvgPIC").

Usage

```
gl.nhybrids(
  gl,
  outfile = "nhyb.txt",
  outpath = tempdir(),
  p0 = NULL,
  p1 = NULL,
```

```
  threshold = 0,
  method = "random",
  plot = TRUE,
  pprob = 0.95,
  nhyb.directory = NULL,
  BurnIn = 10000,
  sweeps = 10000,
  GtypFile = "TwoGensGtypFreq.txt",
  AFPriorFile = NULL,
  PiPrior = "Jeffreys",
  ThetaPrior = "Jeffreys",
  verbose = NULL
)
```

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]
threshold	– 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]
plot	– if TRUE, a plot of the frequency of homozygous reference, heterozygotes and homozygous alternate (SNP) is produced for the F1 individuals [default TRUE, applies only if both parental populations are specified]
pprob	– threshold level for assignment to likelihood bins [default 0.95, used only if plot=TRUE]
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]
verbose	– verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log ; 3, progress and results summary; 5, full report [default 2 or as specified using gl.set.verbosity]

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. threshold = 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 (default="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.

The F1 individuals should be homozygous at all loci for which the parental populations are fixed and different, assuming parental populations have been specified. Sampling errors can result in this not being the case, especially where the sample sizes for the parental populations are small. Alternatively, the threshold for posterior probabilities used to determine assignment (pprob) or the definition of a fixed difference (threshold) may be too lax. To assess the error rate in the determination of assignment of F1 individuals, a plot of the frequency of homozygous reference, heterozygotes and homozygous alternate (SNP) can be produced by setting plot=TRUE (the default).

Value

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

Author(s)

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

Examples

```
## Not run:
m <- gl.nhybrids(testset.gl, outfile="nhyb.txt",
p0=NULL, p1=NULL,
nhyb.directory="C:/workspace/R/NewHybsPC",
BurnIn=100,
sweeps=100,
verbose=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 *Ordination applied to genotypes in a genlight object (PCA), in an fd object, or to a distance matrix (PCoA)*

Description

This function takes the genotypes for individuals and undertakes a Pearson Principal Component analysis (PCA) on SNP or Tag P/A (SilicoDArT) data; it undertakes a Gower Principal Coordinate analysis (PCoA) if supplied with a distance matrix. Technically, any distance matrix can be represented in an ordinated space using PCoA.

Usage

```
gl.pcoa(
  x,
  nfactors = 5,
  correction = NULL,
  parallel = FALSE,
  n.cores = 16,
  verbose = NULL
)
```

Arguments

x	– name of the genlight object or fd object containing the SNP data, or a distance matrix of type dist [required]
nfactors	– number of axes to retain in the output of factor scores.
correction	Method applied to correct for negative eigenvalues, either 'lingoes' or 'cailliez' [Default NULL]
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]

verbose – verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log ; 3, progress and results summary; 5, full report [default 2 or as specified using gl.set.verbosity]

Details

The function is essentially a wrapper for glPca adegenet or pcoa {ape} with default settings apart from those specified as parameters in this function.

While, technically, any distance matrix can be represented in an ordinated space, the representation will not typically be exact.There are three major sources of stress in a reduced-reprentation of distances or dissimilarities among entities using PCA or PCoA. By far the greatest source comes from the decision to select only the top two or three axes from the ordinated set of axes derived from the PCA or PCoA. The representation of the entities such a heavily reduced space will not faithfully represent the distances in the input distance matrix simply because of the loss of information in deeper informative dimensions. For this reason, it is not sensible to be too precious about managing the other two sources of stress in the visual representation.

The measure of distance between entities in a PCA is the Pearson Correlation Coefficent, essentially a standardized Euclidean distance. This is both a metric distance and a Euclidean distance. In PCoA, the second source of stress is the choice of distance measure or dissimilarity measure. While any distance or dissimilarity matrix can be represented in an ordinated space, the distances between entities can befaithfully represented in that space (that is, without stress) only if the distances are metric. Furthermore, for distances between entities to be faithfully represented in a rigid Cartesian space, the distance measure needs to be Euclidean. If this is not the case, the distances between the entities in the ordinated visualized space will not exactly represent the distances in the input matrix (stress will be non-zero). This source of stress will be evident as negative eigenvalues in the deeper dimensions.

A third source of stress arises from having a sparse dataset, one with missing values. This affects both PCA and PCoA. If the original data matrix is not fully populated, that is, if there are missing values, then even a Euclidean distance matrix will not necessarily be 'positive definite'. It follows that some of the eigenvalues may be negative, even though the distance metric is Euclidean. This issue is exacerbated when the number of loci greatly exceeds the number of individuals, as is typically the case when working with SNP data. The impact of missing values can be minimized by stringently filtering on Call Rate, albeit with loss of data. An alternative is given in a paper "Honey, I shrunk the sample covariance matrix" and more recently by Ledoit and Wolf (2018), but their approach has not been implemented here.

The good news is that, unless the sum of the negative eigenvalues, arising from a non-Euclidean distance measure or from missing values, approaches those of the final PCA or PCoA axes to be displayed, the distortion is probably of no practical consequence and certainly not comparable to the stress arising from selecting only two or three final dimensions out of several informative dimensions for the visual representation.

Two diagnostic plots are produced. The first is a Scree Plot, showing the percentage variation explained by each of the PCA or PCoA axes, for those axes that explain more than the original variables (loci) on average. That is, only informative axes are displayed. The scree plot informs the number of dimensions to be retained in the visual summaries. As a rule of thumb, axes with more than 10

The second graph shows the distribution of eigenvalues for the remaining uninformative (noise) axes, including those with negative eigenvalues. Action is recommended (verbose >= 2) if the

negative eigenvalues are dominant, their sum approaching in magnitude the eigenvalues for axes selected for the final visual solution.

Output is a glPca object conforming to adegenet::glPca but with only the following retained. $call The call that generated the PCA/PCoA $eig Eigenvalues – All eigenvalues (positive, null, negative). $scores Scores (coefficients) for each individual $loadings Loadings of each SNP for each principal component

PCA was developed by Pearson (1901) and Hotelling (1933), whilst the best modern reference is Jolliffe (2002). PCoA was developed by Gower (1966) while the best modern reference is Legendre & Legendre (1998).

Value

An object of class pcoa containing the eigenvalues and factor scores

Author(s)

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

References

Cailliez, F. (1983) The analytical solution of the additive constant problem. Psychometrika, 48, 305-308. Gower, J. C. (1966) Some distance properties of latent root and vector methods used in multivariate analysis. Biometrika, 53, 325-338. Hotelling, H., 1933. Analysis of a complex of statistical variables into Principal Components. Journal of Educational Psychology 24:417-441, 498-520. Jolliffe, I. (2002) Principal Component Analysis. 2nd Edition, Springer, New York. Ledoit, O. and Wolf, M. (2018). Analytical nonlinear shrinkage of large-dimensional covariance matrices. University of Zurich, Department of Economics, Working Paper No. 264, Revised version. Available at SSRN: https://ssrn.com/abstract=3047302 or http://dx.doi.org/10.2139/ssrn.3047302 Legendre, P. and Legendre, L. (1998). Numerical Ecology, Volume 24, 2nd Edition. Elsevier Science, NY. Lingoes, J. C. (1971) Some boundary conditions for a monotone analysis of symmetric matrices. Psychometrika, 36, 195-203. Pearson, K. (1901). On lines and planes of closest fit to systems of points in space. Philosophical Magazine. Series 6, vol. 2, no. 11, pp. 559-572.

Examples

```
fd <- gl.fixed.diff(testset.gl)
fd <- gl.collapse(fd)
pca <- gl.pcoa(fd)
gl.pcoa.plot(pca,fd)
```

---

gl.pcoa.plot *Bivariate plot of the results of an ordination generated using gl.pcoa()*

Description

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

Usage

```
gl.pcoa.plot(
  glPca,
  x,
  scale = FALSE,
  ellipse = FALSE,
  p = 0.95,
  labels = "pop",
  theme_plot = 4,
  as.pop = NULL,
  hadjust = 1.5,
  vadjust = 1,
  xaxis = 1,
  yaxis = 2,
  plot.out = TRUE,
  verbose = NULL
)
```

Arguments

glPca	Name of the PCA or PCoA object containing the factor scores and eigenvalues [required]
x	Name of the genlight object or fd object containing the SNP genotypes or a genlight object containing the Tag P/A (SilicoDArT) genotypes or the Distance Matrix used to generate the ordination [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"]
theme_plot	Theme for the plot. See Details for options [default 4]
as.pop	– assign another metric to represent populations for the plot [default NULL]
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)
plot.out	If TRUE, returns a plot object compatable with ggplot, otherwise returns a dataframe [default TRUE]
verbose	– verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log ; 3, progress and results summary; 5, full report [default 2 or as specified using gl.set.verbosity]

Details

The factor scores are taken from the output of gl.pcoa() 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 (e.g. 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. Identification of points is by moving the mouse over them. Refer to the plotly package for further information.

If plot.out=TRUE, returns an object of class ggplot so that layers can subsequently be added; if plot.out=FALSE, returns a dataframe with the individual labels, population labels and PCOA scores for subsequent plotting by the user with ggplot or other plotting software.

The themes available to format the plot are the following: theme_minimal[1], theme_classic[2],theme_bw[3],theme_gray[4],t
Examples of these themes can be consulted in https://ggplot2.tidyverse.org/reference/ggtheme.html and https://yutannihilation.github.io/allYourFigureAreBelongToUs/ggthemes/

Value

A plot of the ordination [plot.out=TRUE] or a dataframe [plot.out=FALSE]

Author(s)

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

Examples

```
if (requireNamespace("directlabels", quietly = TRUE)) {
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")
pca<-gl.pcoa(gl,nfactors=5)
gl.pcoa.plot(pca, gl, ellipse=TRUE, p=0.99, labels="pop",hadjust=1.5,
 vadjust=1)
if (requireNamespace("plotly", quietly = TRUE)) {
#interactive plot to examine labels
gl.pcoa.plot(pca, gl, labels="interactive")
}
}
```

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, zoomed in and zoomed out with the mouse to examine the structure.

Usage

```
gl.pcoa.plot.3d(
  glPca,
  x,
  xaxis = 1,
  yaxis = 2,
  zaxis = 3,
  radius = 8,
  verbose = NULL
)
```

Arguments

glPca	– name of the glPca object containing the factor scores and eigenvalues [required]
x	– name of the genlight object or fd object from which the PCoA was generated [required]
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]
radius	– size of the points [default 8]
verbose	– verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log ; 3, progress and results summary; 5, full report [default 2 or as specified using gl.set.verbosity]

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().

Value

NULL, plots an interactive 3D plot of the ordination in a separate window

Author(s)

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

Examples

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

---

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, verbose = 0)
```

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]
verbose	– verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log ; 3, progress and results summary; 5, full report [default 2 or as specified using gl.set.verbosity]

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 (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 object {adegenet} and generate a table of allele frequencies for each population and locus

Usage

```
gl.percent.freq(x, verbose = NULL)
```

Arguments

x	– name of the genlight object containing the SNP or Tag P/A (SilicoDArT) data [required]
verbose	– verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log ; 3, progress and results summary; 5, full report [default 2 or as specified using gl.set.verbosity]

Value

A matrix with allele (SNP data) or presence/absence frequencies (Tag P/A data) broken down by population and locus

Author(s)

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

Examples

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

`gl.play.history` *Replays the history and applies it to a genlight object*

Description

Replays the history and applies it to a genlight object

Usage

```
gl.play.history(x, history = NULL, verbose = 0)
```

Arguments

`x`	– a genlight object [with a history slot [optional]]
`history`	– [optional]. If no history is provided the complete history of x is used (recreating the identical object x). If history is a vector it indicates which which part of the history of x is used [`c(1,3,4)` uses the first, third and forth entry from `x@other$history`]. Or a simple link to a history slot of another genlight object (e.g. codex2@other$history[c(1,4,5)]).
`verbose`	[default 0]. If set to one then history commands are printed, which may facilitate reading the output.

Details

This function basically allows to create a "template history" (=set of filters) and apply them to any other genlight object. Histories can also be saved and loaded (see. gl.save.history and gl.load.history).

Value

returns a genlight object that was created by replaying the provided applied to the genlight object x. Please note you can "mix" histories or part of them and apply them to different genlight objects. If the history does not contain `gl.read.dart`, histories of x and history are concatenated.

Author(s)

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

Examples

```
dartfile <- system.file("extdata","testset_SNPs_2Row.csv", package="dartR")
metadata <- system.file("extdata","testset_metadata.csv", package="dartR")
gl <- gl.read.dart(dartfile, ind.metafile = metadata, probar=FALSE)
gl2 <- gl.filter.callrate(gl, method="loci", threshold=0.9)
gl3 <- gl.filter.callrate(gl2, method="ind", threshold=0.95)
#Now "replay" part of the history "onto" another genlight object
bc.fil <- gl.play.history(gl.compliance.check(bandicoot.gl),
history=gl3@other$history[c(2,3)], verbose=1)
gl.print.history(bc.fil)
```

`gl.plot` *Plotting genlight object as a smear plot (loci by individuals color coded for scores of 0, 1, 2 and NA)*

Description

This function is based on the glPlot function from adegenet. It adds the option to put labels on the individuals and scales them accordingly. If there are too many individuals, it is best to use labels=FALSE.

For arguments please refer to the original adegenet function ?glPlot.

Usage

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

Arguments

`x`	– a genlight object [required]
`labels`	– if TRUE, individual labels are added
`indlabels`	– labels for individuals [default = first 8 letters from indNames]
`col`	– optional color vector (see ?glPlot) [default NULL]
`legend`	– if TRUE, a legend will be added [default = TRUE]
`posi`	– position of the legend [default = "bottomleft"]
`bg`	– background color of the legend [default transparent white]
`verbose`	– verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log ; 3, progress and results summary; 5, full report [default NULL]
`...`	— additional arguments passed to glPlot function.

Examples

```
gl.plot(bandicoot.gl[1:30,])
gl.plot(bandicoot.gl[1:30,])
gl.plot(bandicoot.gl[1:10,],labels=TRUE)
```

gl.plot.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.plot.heatmap(D, verbose = NULL)
```

Arguments

D	– name of the distance matrix or class fd object [required]
verbose	– 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) as a wrapper for pheatmap by Raivo Kolde.

Examples

```
gl <- testset.gl[1:10,]
D <- dist(as.matrix(gl),upper=TRUE,diag=TRUE)
gl.plot.heatmap(D)
D2 <- gl.dist.pop(testset.gl)
gl.plot.heatmap(D2)
D3 <- gl.fixed.diff(testset.gl)
gl.plot.heatmap(D3)
```

gl.plot.network *Represents a distance or dissimilarity matrix as a network*

Description

This script takes a distance matrix generated by dist() 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.plot.network(
  D,
  x = NULL,
  method = "fr",
  node.size = 3,
  node.label = FALSE,
  node.label.size = 0.7,
  node.label.color = "black",
  alpha = 0.005,
  title = "Network based on genetic distance",
  verbose = 2
)
```

Arguments

D	– a distance or dissimilarity matrix generated by dist() or gl.dist() [required]
x	– genlight object from which the D matrix was generated [optional, default=NULL]
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: 0.005]
title	– title for the plot [default: "Network based on G-matrix of genetic relatedness"]
verbose	– 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 (Post to https://groups.google.com/d/forum/dartr)

Examples

```
D <- gl.grm(bandicoot.gl)
gl.plot.network(D,bandicoot.gl)
```

---

gl.print.history *Prints history of a genlight object*

Description

Prints history of a genlight object

Usage

```
gl.print.history(x = NULL, history = NULL)
```

Arguments

x	– a genlight object (with history [optional])
history	– [optional] either a link to a history slot (gl\@other$history), or a vector indicating which part of the history of x is used [c(1,3,4) uses the first, third and forth entry from x\@other$history]. If no history is provided the complete history of x is used (recreating the identical object x).

Value

prints a table with all history records. Currently the style cannot be changed.

Author(s)

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

Examples

```
dartfile <- system.file("extdata","testset_SNPs_2Row.csv", package="dartR")
metadata <- system.file("extdata","testset_metadata.csv", package="dartR")
gl <- gl.read.dart(dartfile, ind.metafile = metadata, probar=FALSE)
gl2 <- gl.filter.callrate(gl, method="loci", threshold=0.9)
gl3 <- gl.filter.callrate(gl2, method="ind", threshold=0.95)
#Now "replay" part of the history "onto" another genlight object
bc.fil <- gl.play.history(gl.compliance.check(bandicoot.gl),
history=gl3@other$history[c(2,3)], verbose=1)
gl.print.history(bc.fil)
```

gl.propShared — *Calculate a similarity(distance) matrix for individuals on the proportion of shared alleles*

Description

This script calculates a individual based distance matrix. It uses an C++ implementation, so package Rcpp needs to be installed and it is therefore really fast (once it has compiled the function after the first run).

Usage

```
gl.propShared(x)
```

Arguments

x	– name of the genlight containing the SNP genotypes [required]

Author(s)

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

Examples

```
#takes some time at the first run of the function...
## Not run:
res <- gl.propShared(bandicoot.gl)
res[1:5,1:7] #show only a small part of the matrix

## End(Not run)
```

gl.read.csv — *Read SNP data from a csv file into a genlight object*

Description

This script takes SNP genotypes from a csv file, combines them with individual and locus metrics and creates a genlight object.

Usage

```
gl.read.csv(
  filename,
  transpose = FALSE,
  ind.metafile = NULL,
  loc.metafile = NULL,
  verbose = NULL
)
```

Arguments

filename	– name of the csv file containing the SNP genotypes [required]
transpose	– if TRUE, rows are loci and columns are individuals [default FALSE]
ind.metafile	– name of the csv file containing the metrics for individuals [optional]
loc.metafile	– name of the csv file containing the metrics for loci [optional]
verbose	– 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 SNP data need to be in one of two forms. SNPs can be coded 0 for homozygous reference, 2 for homozygous alternate, and 1 for heterozygous with NA for missing values; or the SNP data can be coded A/A, A/C, C/T, G/A etc, with -/- as missing. Other formats will throw an error.

The SNP data need to be individuals as rows,labelled, and loci as columns, also labelled. If the orientation is individuals as columns and loci by rows, then set transpose=TRUE.

The individual metrics need to be in a csv file, with headings, with a mandatory id column corresponding exactly to the individual identity labels provided with the SNP data and in the same order.

The locus metadata needs to be in a csv file with headings, with a mandatory column headed AlleleID corresponding exactly to the locus identity labels provided with the SNP data and in the same order.

Note that the locus metadata will be complemented by calculable statistics corresponding to those that would be provided by Diversity Arrays Technology (e.g. CallRate)

Value

a genlight object with the SNP data and associated metadata included.

Author(s)

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

---

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,
  recalc = FALSE,
  mono.rm = FALSE,
  nas = "-",
  topskip = NULL,
  lastmetric = "RepAvg",
  covfilename = NULL,
  probar = FALSE,
  verbose = NULL
)
```

Arguments

filename	file containing the SNP data (csv file) [required]
ind.metafile	file that contains additional information on individuals [required]
recalc	force the recalculation of locus metrics, in case individuals have been manually deleted from the input csv file [FALSE]
mono.rm	force the removal of monomorphic loci (including all NAs), in case individuals have been manually deleted from the input csv file [FALSE]
nas	a character specifying NAs [nas = '-']
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.
covfilename	use ind.metafile parameter [deprectiated, NULL]
probar	show progress bar
verbose	– verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log ; 3, progress and results summary; 5, full report [default 2, or as set by gl.set.verbose()]

Value

a dart genlight object that contains individual metrics [if data were provided] and locus metrics [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 `utils.read.dart`.

Examples

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

```
metadata <- system.file("extdata","testset_metadata.csv", package="dartR")
gl <- gl.read.dart(dartfile, ind.metafile = metadata, probar=TRUE)
}
```

`gl.read.silicodart` *Import presence/absence data from SilicoDArT to genlight {agegenet} format (ploidy=1)*

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 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(
  filename,
  ind.metafile = NULL,
  nas = "-",
  topskip = NULL,
  lastmetric = "Reproducibility",
  probar = TRUE,
  verbose = NULL
)
```

Arguments

`filename`	– name of csv file containing the SilicoDArT data [required]
`ind.metafile`	– name of csv file containing metadata assigned to each entity (individual) [default NULL]
`nas`	– missing data character [default "-"]
`topskip`	– number of rows to skip before the header row (containing the specimen identities) [optional]
`lastmetric`	– specifies the last non genetic column (Default is "Reproducibility"). 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. [default Reproducibility]
`probar`	show progress bar
`verbose`	– verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log ; 3, progress and results summary; 5, full report [default 2, or as set by gl.set.verbose()]

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 `genlight` with ploidy set to 1, containing the presence/absence data, and locus and individual metadata

Author(s)

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

Examples

```
silicodartfile <- system.file("extdata","testset_SilicoDArT.csv", package="dartR")
metadata <- system.file("extdata",ind.metafile ="testset_metadata_silicodart.csv", package="dartR")
testset.gs <- gl.read.silicodart(filename = silicodartfile, ind.metafile = metadata)
```

---

gl.read.vcf *Converts a vcf file into a genlight object*

Description

This function needs package vcfR, please install it. The converted genlight object does not have individual metrics. You need to add them 'manually' to the other$ind.metrics slot.

Usage

```
gl.read.vcf(vcffile, verbose = 2)
```

Arguments

vcffile	– a vcf file (works only for diploid data)
verbose	set to 2

Value

A genlight object.

Author(s)

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

Examples

```
## Not run:
test <- gl.read.vcf("your_data.vcf")

## End(Not run)
```

---

gl.reassign.pop *Assign a individual metric as pop in a genlight {adegenet} object*

Description

Individuals are assigned to populations based on the individual/sample/specimen metrics file (csv) used with gl.read.dart().

Usage

```
gl.reassign.pop(x, as.pop, verbose = NULL)
```

Arguments

x	– name of the genlight object containing SNP genotypes [required]
as.pop	– specify the name of the individual metric to set as the pop variable. [required]
verbose	– verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log ; 3, progress and results summary; 5, full report [default 2 or as specified using gl.set.verbosity]

Details

One might want to define the population structure in accordance with another classification, such as using an individual metric (e.g. sex, male or female). This script discards the current population assignments and replaces them with new population assignments defined by a specified individual metric.

The script returns a genlight object with the new population assignments Note that the original population assigments are lost.

Value

A genlight object with the reassigned populations

Author(s)

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

Examples

```
# SNP data
   popNames(testset.gl)
   gl <- gl.reassign.pop(testset.gl, as.pop='sex',verbose=3)
   popNames(gl)
# Tag P/A data
   popNames(testset.gs)
   gs <- gl.reassign.pop(testset.gs, as.pop='sex',verbose=3)
   popNames(gs)
```

---

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, mono.rm = FALSE, verbose = NULL)
```

Arguments

x	– name of the genlight object containing SNP genotypes [required]
mono.rm	– if TRUE, removes monomorphic loci [default FALSE]
verbose	– verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log ; 3, progress and results summary; 5, full report [default 2 or as specified using gl.set.verbosity]

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

See Also

gl.filter.monomorphs

Examples

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

---

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

Description

This script recodes individual labels and/or deletes individuals from a DaRT genlight SNP file based on a lookup table provided as a csv file.

Usage

```
gl.recode.ind(x, ind.recode, recalc = FALSE, mono.rm = FALSE, verbose = NULL)
```

Arguments

x	– name of the genlight object containing SNP genotypes [required]
ind.recode	– name of the csv file containing the individual relabelling [required]
recalc	– if TRUE, recalculate the locus metadata statistics if any individuals are deleted in the filtering [default FALSE]
mono.rm	– if TRUE, remove monomorphic loci [default FALSE]
verbose	– verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log ; 3, progress and results summary; 5, full report [default 2 or as specified using gl.set.verbosity]

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. When caution needs to be exercised because of the potential for breaking the "chain of evidence" associated with the samples, recoding individuals using a recode table (csv) can provide a clear record of the changes.

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 incorrect by the deletion of individuals from the dataset.

The script returns a genlight object with the new individual labels, the monomorphic loci optionally removed and the optionally recalculated locus metadata.

Value

A genlight or genind object with the recoded and reduced data

Author(s)

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

See Also

gl.filter.monomorphs for filtering monomorphs, gl.recalc.metrics for recalculating locus metrics, gl.recode.pop for recoding populations

Examples

```
file <- system.file("extdata","testset_pop_recode.csv", package="dartR")
#gl <- gl.recode.ind(testset.gl, ind.recode=file, verbose=0)
```

---

gl.recode.pop *Recode population assignments in a genlight 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, verbose = NULL)
```

Arguments

x	– name of the genlight object containing the SNP 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 FALSE]
mono.rm	– Remove monomorphic loci [default FALSE]
verbose	– verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log ; 3, progress and results summary; 5, full report [default 2 or as specified using gl.set.verbosity]

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

See Also

gl.filter.monomorphs

Examples

```
mfile <- system.file("extdata", "testset_pop_recode.csv", package="dartR")
nPop(testset.gl)
gl <- gl.recode.pop(testset.gl, pop.recode=mfile, verbose=3)
```

---

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(x, plot = TRUE, verbose = NULL)
```

Arguments

x	– name of the genlight object containing the SNP or presence/absence data [required]
plot	– if TRUE, histograms of base composition are produced [default TRUE]
verbose	– verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log ; 3, progress and results summary; 5, full report [default 2 or as specified using gl.set.verbosity]

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.

For presence/absence data (SilicoDArT), it is not possible to count transitions and tranversions or tv/ts ratio because the SNP data is not available, only a single sequence tag.

Value

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

returns a named vector of base frequencies and the transversion and transitions. I also returns the plot as an ggplot object, which can be further customised. See example.

Author(s)

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

Examples

```
# SNP data
  out <- gl.report.bases(testset.gl)
  out$freq
  out$plotbases
  out$plottvts
# Tag P/A data
  out <- gl.report.bases(testset.gs)
  out
```

---

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",
  boxplot = "adjusted",
  range = 1.5,
  verbose = NULL
)
```

Arguments

x	– name of the genlightobject containing the SNP or presence/absence (SilicoDArT) data [required]
method	specify the type of report by locus (method="loc") or individual (method="ind") [default method="loc"]
boxplot	– if 'standard', plots a standard box and whisker plot; if 'adjusted', plots a boxplot adjusted for skewed distributions [default 'adjusted']
range	– specifies the range for delimiting outliers [default = 1.5 interquartile ranges]
verbose	– verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log ; 3, progress and results summary; 5, full report [default 2 or as specified using gl.set.verbosity]

Details

Tag Presence/Absence datasets (SilicoDArT) have missing values where it is not possible to determine reliably if there the sequence tag can be called at a particular locus.

The minimum, maximum and mean call rate are provided. Output also is a histogram of read depth, accompanied by a box and whisker plot.

Refer to Tukey (1977, Exploratory Data Analysis. Addison-Wesley) for standard Box and Whisker Plots and Hubert & Vandervieren (2008), An Adjusted Boxplot for Skewed Distributions, Computational Statistics & Data Analysis 52:5186-5201) for adjusted Box and Whisker Plots.

Value

returns a tabulation of CallRate against Threshold

Author(s)

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

Examples

```
# SNP data
  out <- gl.report.callrate(testset.gl)
# Tag P/A data
  out <- gl.report.callrate(testset.gs)
```

`gl.report.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.report.diversity(
  gl,
  spectrumplot = TRUE,
  confiplot = FALSE,
  pbar = TRUE,
  table = "DH",
  verbose = NULL
)
```

Arguments

`gl`	genlight object containing the SNP genotypes [required]
`spectrumplot`	switch to provide a plot [default TRUE]
`confiplot`	switch if confidence intervals (1 sd) should be drawn [default FALSE]
`pbar`	report on progress. Silent if set to FALSE. [default 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.
`verbose`	– verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log ; 3, progress and results summary; 5, full report [default 2 or as specified using gl.set.verbosity]

Value

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

Author(s)

Bernd Gruber (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

Examples

```
div <- gl.report.diversity(bandicoot.gl, spectrumplot = TRUE, table = FALSE, pbar=FALSE)
div$zero_H_alpha
div$two_H_beta
names(div)
```

---

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(
  x,
  rs = 5,
  boxplot = "adjusted",
  range = 1.5,
  threshold = 3,
  taglength = 69,
  probar = FALSE,
  verbose = 2
)
```

Arguments

x	– name of the genlight object containing the SNP data [required]
rs	– number of bases in the restriction enzyme recognition sequence [default 5]
boxplot	– if 'standard', plots a standard box and whisker plot; if 'adjusted', plots a box-plot adjusted for skewed distributions [default 'adjusted']
range	– specifies the range for delimiting outliers [default = 1.5 interquartile ranges]
threshold	minimum acceptable base pair difference for display on the whisker plot and histogram [default 3 bp]
taglength	– typical length of the sequence tags [default 69]
probar	– if TRUE, then a progress bar is desplayed on long loops [default TRUE]
verbose	– verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log ; 3, progress and results summary; 5, full report [default 2 or as specified using gl.set.verbosity]

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

A histogram and whiskerplot can be requested. Both display a user specified value for the minumum acceptable Hamming distance.

Value

Tabulation of loc that will be lost on filtering, against values of the threshold

Author(s)

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

Examples

```
out <- gl.report.hamming(testset.gl)
```

gl.report.heterozygosity

Reports observed and expected heterozygosity by population or by individual from SNP data.

Description

Calculates the observed and expected heterozygosities for each population (method="pop") or the observed heterozyosity for each individual (method="ind") in a genlight object.

Usage

```
gl.report.heterozygosity(
  x,
  method = "pop",
  n.invariant = 0,
  plot = TRUE,
  boxplot = "adjusted",
  range = 1.5,
  cex.labels = 0.7,
  verbose = NULL
)
```

Arguments

x	– a genlight object containing the SNP genotypes [Required]
method	– calculate heterozygosity by population (method='pop') or by individual (method='ind') [default 'pop']
n.invariant	– an estimate of the number of invariant sequence tags used to adjust the heterozygosity rate [default 0]
plot	– if TRUE, plots barcharts of observed and expected heterozygosity for populations [TRUE]
boxplot	– if 'standard', plots a standard box and whisker plot; if 'adjusted', plots a boxplot adjusted for skewed distributions [default 'adjusted']
range	– specifies the range for delimiting outliers [default = 1.5 interquartile ranges]
cex.labels	– sets the size of the population labels [default 0.7]
verbose	– verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log ; 3, progress and results summary; 5, full report [default 2 or as specified using gl.set.verbosity]

Details

Observed heterozygosity for a population takes the proportion of heterozygous loci for each individual then averages over the individuals in that population. The calculations take into account missing values.

Expected heterozygosity for a population takes the expected proportion of heterozygotes, that is, expected under Hardy-Weinberg equilibrium, for each locus, then averages this across the loci for an average estimate for the population. The calculations of expected heterozygosity use the unbiassed estimates of Nei, M. (1987) Molecular evolutionary genetics. New York: Columbia University Press.

Output for method="pop" is an ordered barchart of observed heterozygosity across populations together with a table of observed and expected heterozygosity by population.

Observed heterozygosity for individuals is calculated as the proportion of loci that are heterozygous for that individual.

Output for method="ind" is a histogram of heterozygosity across individuals. The histogram is accompanied by a box and whisker plot presented either in standard (boxplot="standard") or adjusted for skewness (boxplot=adjusted).

Refer to Tukey (1977, Exploratory Data Analysis. Addison-Wesley) for standard Box and Whisker Plots and Hubert & Vandervieren (2008), An Adjusted Boxplot for Skewed Distributions, Computational Statistics & Data Analysis 52:5186-5201) for adjusted Box and Whisker Plots.

Finally, the loci that are invariant across all individuals in the dataset (that is, across populations), is typically unknown. This can render estimates of heterozygosity analysis specific, and so it is not valid to compare such estimates across species or even across different analyses. This is a similar problem faced by microsatellites. If you have an estimate of the number of invariant sequence tags (loci) in your data, such as provided by gl.report.secondaries, you can specify it with the n.invariant parameter to standardize your estimates of heterozygosity.

Value

returns a dataframe containing population labels, heterozygosities and sample sizes

Author(s)

Bernd Gruber, Arthur Georges and Renee Catullo (Post to https://groups.google.com/d/forum/dartr)

Examples

```
out <- gl.report.heterozygosity(testset.gl,verbose=3)
out <- gl.report.heterozygosity(testset.gl,method='ind',verbose=3)
```

---

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 utils.prob.hwe() as developed by Wigginton et al. (2005).

Usage

```
gl.report.hwe(
  x,
  subset = "each",
  plot = FALSE,
  method = "ChiSquare",
  alpha = 0.05,
  bonf = TRUE,
  verbose = NULL
)
```

Arguments

x	– a genlight object containing the SNP genotypes [Required]
subset	– either, list populations to combine in the analysis | each | all [Default "each"]
plot	– if TRUE, will produce a Ternary Plot(s) [default FALSE]
method	– for determining the statistical signicance in the ternary plot: ChiSquare (with continuity correction) | Fisher [default "ChiSquare"]
alpha	– level of significance for testing [default 0.05]
bonf	– if TRUE, Bonferroni correction will be applied to the level of significance [default TRUE]
verbose	– verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log ; 3, progress and results summary; 5, full report [default 2 or as specified using gl.set.verbosity]

Details

Tests are applied to each locus across all populations pooled (subset="all"), to each locus considered within each population treated separately (subset="each") or to each locus within selected populations pooled (subset=c("pop1","pop2")). Tests for HWE are only valid if there is no population substructure (assuming random mating), and the tests have sufficient power only when there is sufficient sample size (say, n individuals > 20). Note also that correction for multiple comparisons is probably required if you wish to place particular importance on one or a few significant departures.

A Ternary Plot is optionally produced – see Graffelman et al.(2008) for further details. Implementation of the Ternary Plot is via package HardyWeinberg (Graffelman (2015). The plot labels loci that depart significantly from HWE as red, and those not showing significant departure as green. Two methods are used to determine significance. ChiSquare (with correction) is traditional but involves approximations; Fisher is computationally more expensive, but applies a Fisher Exact Test of departure from HWE.

Value

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

References

Wigginton, J.E., Cutler, D.J., & Abecasis, G.R. (2005). A Note on Exact Tests of Hardy-Weinberg Equilibrium. American Journal of Human Genetics 76:887-893.

Graffelman, J. & Morales-Camarena, J. (2008). Graphical tests for Hardy-Weinberg equilibrium based on the ternary plot. Human Heredity 65:77-84.

Graffelman, J. (2015). Exploring Diallelic Genetic Markers: The HardyWeinberg Package. Journal of Statistical Software 64:1-23.

Examples

```
list <- gl.report.hwe(testset.gl,subset=c("EmmacMaclGeor", "EmmacCoopCully"),plot=TRUE,bonf=FALSE)
gl.report.hwe(testset.gl,subset=c("EmmacCoopCully"), plot=TRUE, verbose=3)
gl.report.hwe(testset.gl,subset="all", plot=TRUE, bonf=FALSE, verbose=3)
gl.report.hwe(testset.gl, subset="each", plot=TRUE, bonf=FALSE)
```

---

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(
  x,
  name = NULL,
  save = TRUE,
  outpath = tempdir(),
  nchunks = 2,
  ncores = 1,
  chunkname = NULL,
  probar = FALSE,
  verbose = NULL
)
```

Arguments

x	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
outpath	folder where chunks and results are saved (if save=TRUE). Default it tempdir()
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]
probar	if TRUE, a progress bar is displayed for long loops [default = TRUE]
verbose	– verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log ; 3, progress and results summary; 5, full report [default 2 or as specified using gl.set.verbosity]

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

---

gl.report.locmetric *Report summary of the slot $other$loc.metrics*

Description

This script uses any field with numeric values stored in $other$loc.metrics to produce summary statistics (mean, minimum, average, percentiles), histograms and boxplots to assist the decision of choosing thresholds for the filter function gl.filter.locmetric(). The fields that are included in dartR, and a short description, are found below. Optionally, the user can also set his/her own field by adding a vector into $other$loc.metrics as shown in the example. You can check the names of all available loc.metrics via: names(gl$other$loc.metrics).

Usage

```
gl.report.locmetric(x, metric, verbose = NULL)
```

Arguments

x	– name of the genlight object containing the SNP or presence/absence (Silico-DArT) data [required]
metric	– name of the metric to be used for filtering [required]
verbose	– verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log ; 3, progress and results summary; 5, full report [default 2 or as specified using gl.set.verbosity]

Details

- SnpPosition - position (zero is position 1) in the sequence tag of the defined SNP variant base - CallRate - proportion of samples for which the genotype call is non- missing (that is, not '- ') - OneRatioRef - proportion of samples for which the genotype score is 0 - OneRatioSnp - proportion of samples for which the genotype score is 2 - FreqHomRef - proportion of samples homozygous for the Reference allele - FreqHomSnp - proportion of samples homozygous for the Alternate (SNP) allele - FreqHets - proportion of samples which score as heterozygous, that is, scored as 1 - PICRef - polymorphism information content (PIC) for the Reference allele - PICSnp - polymorphism information content (PIC) for the SNP - AvgPIC - average of the polymorphism information content (PIC) of the Reference and SNP alleles - AvgCountRef - sum of the tag read counts for all samples, divided by the number of samples with non-zero tag read counts, for the Reference allele row - AvgCountSnp - sum of the tag read counts for all samples, divided by the number of samples with non-zero tag read counts, for the Alternate (SNP) allele row - RepAvg - proportion of technical replicate assay pairs for which the marker score is consistent - rdepth - read depth

Value

returns a tabulation of locmetrics against different thresholds

Author(s)

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

Examples

```
# adding dummy data
test <- testset.gl
test$other$loc.metrics$test <- 1:nLoc(test)
# SNP data
out <- gl.report.locmetric(test,metric="test")

# adding dummy data
test.gs <- testset.gs
test.gs$other$loc.metrics$test <- 1:nLoc(test.gs)
# Tag P/A data
out <- gl.report.locmetric(test.gs,metric="test")
```

`gl.report.maf` *Report minor allele frequency (MAF) for each locus in a SNP dataset*

Description

This script provides summary histograms of MAF for each population in the dataset as a basis for decisions on filtering.

Usage

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

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]
`verbose`	– verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log ; 3, progress and results summary; 5, full report [default 2 or as specified using gl.set.verbosity]

Author(s)

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

Examples

```
gl.report.maf(testset.gl)
```

`gl.report.monomorphs` *Report monomorphic loci*

Description

This script reports the number of monomorphic locim and those with all NAsm from a genlight {adegenet} object

Usage

```
gl.report.monomorphs(x, verbose = NULL)
```

Arguments

x	– name of the input genlight object [required]
verbose	– verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log ; 3, progress and results summary; 5, full report [default 2 or as specified using gl.set.verbosity]

Details

A DArT dataset will not have monomorphic loci, but they can arise, along with loci that are scored all NA, when populations or individuals are deleted. Retaining monomorphic loci unnecessarily increases the size of the dataset and will affect some calculations.

Note that for SNP data, NAs likely represent null alleles; in tag presence/absence data, NAs represent missing values (presence/absence could not be reliably scored)

Author(s)

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

Examples

```
# SNP data
  out <- gl.report.monomorphs(testset.gl)
# SilicoDArT data
  out <- gl.report.monomorphs(testset.gs)
```

`gl.report.overshoot` *Reports loci for which the SNP has been trimmed from the sequence tag along with the adaptor*

Description

This function checks the position of the SNP within the trimmed sequence tag and identifies those for which the SNP position is outside the trimmed sequence tag. This can happen, rarely, when the sequence containing the SNP resembles the adaptor.

Usage

```
gl.report.overshoot(x, verbose = NULL)
```

Arguments

`x`	– name of the genlight object [required]
`verbose`	– verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log ; 3, progress and results summary; 5, full report [default 2 or as specified using gl.set.verbosity]

Details

The SNP genotype can still be used in most analyses, but functions like gl2fasta() will present challenges if the SNP has been trimmed from the sequence tag.

Value

returns names of the recalcitrant loci

Author(s)

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

Examples

```
out <- gl.report.overshoot(testset.gl)
```

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

Description

This function reports private alleles in one population compared with a second population, for all populations taken pairwise. It also reports a count of fixed allelic differences and the mean absolute allele frequency differences between pairs of populations.

Usage

```
gl.report.pa(gl1, gl2 = NULL, verbose = NULL)
```

Arguments

`gl1`	– name of the genlight object containing the SNP data [required]
`gl2`	– if two seperate genlight objects are to be compared this can be provided here [default NULL]
`verbose`	– verbosity: 0, silent, fatal errors only; 1, flag function begin and end; 2, progress log ; 3, progress and results summary; 5, full report [default 2 or as specified using gl.set.verbosity]

Details

Note that the number of paired alleles between to populations is not a symmetric dissimilarity measure.

if no gl2 is provided, the function uses the pop(gl) hierachy to determine pairs of population, otherwise it runs a single comparison between gl1 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 alleles

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

returns a data.frame. 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

```
out <- gl.report.pa(testset.gl[1:20,])
```

---

gl.report.parent.offspring

Identify putative parent offspring within a population

Description

This script examines the frequency of pedigree inconsistent loci, that is, those loci that are homozygotes in the parent for the reference allele, and homozygous in the offspring for the alternate allele. This condition is not consistent with any pedigree, regardless of the (unkonwn) genotype of the other parent. The pedigree inconsistent loci are counted as an indication of whether or not it is reasonable to propose the two individuals are in a parent-offspring relationship.

Usage

```
gl.report.parent.offspring(
  x,
  min.rdepth = 12,
  min.reproducibility = 1,
  range = 1.5,
  verbose = NULL
)
```

Arguments

x	Name of the genlight object containing the SNP genotypes [required]
min.rdepth	Minimum read depth to include in analysis [default = 12]
min.reproducibility	Minimum reproducibility to include in analysis [default = 1]
range	– specifies the range to extend beyond the interquartile range for delimiting outliers [default = 1.5 interquartile ranges]
verbose	– verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log ; 3, progress and results summary; 5, full report [default 2 or as specified using gl.set.verbosity]

Details

Obviously, if the two individuals are in a parent offspring relationship, the true number of pedigree inconsistent loci should be zero, but SNP calling is not infallible. Some loci will be mis-called. The problem thus becomes one of determining if the two focal individuals have a count of pedigree inconsistent loci less than would be expected of typical unrelated individuals. There are some quite sophisticated software packages available to formally apply likelihoods to the decision, but we use a simple outlier comparison.

To reduce the frequency of mis-calls, and so emphasise the difference between true parent-offspring pairs and unrelated pairs, the data can be filtered on read depth. Typically minimum read depth is set to 5x, but you can examine the distribution of read depths with gl.report.rdepth() and push this up with an acceptable loss of loci. 12x might be a good minimum for this particular analysis. It is sensible also to push the minimum reproducibility up to 1, if that does not result in an unacceptable loss of loci.

Note that the null expectation is not well defined, and the power reduced, if the population from which the putative parent-offspring pairs are drawn contains many sibs. Note also that if an individual has been genotyped twice in the dataset, the replicate pair will be assessed by this script as being in a parent-offspring relationship.

Value

A set of individuals in parent-offspring relationship

---

gl.report.rdepth *Report summary of Read Depth for each locus*

Description

SNP datasets generated by DArT report AvgCountRef and AvgCountSnp as counts of sequence tags for the reference and alternate alleles respectively. These can be used to backcalculate Read Depth. Fragment presence/absence datasets as provided by DArT (SilicoDArT) provide Average Read Depth and Standard Deviation of Read Depth as stanard columns in their report.

Usage

```
gl.report.rdepth(x, boxplot = "adjusted", range = 1.5, verbose = NULL)
```

Arguments

x	– name of the genlight object containing the SNP data [required]
boxplot	– if 'standard', plots a standard box and whisker plot; if 'adjusted', plots a boxplot adjusted for skewed distributions [default 'adjusted']
range	– specifies the range for delimiting outliers [default = 1.5 interquartile ranges]
verbose	– verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log ; 3, progress and results summary; 5, full report [default 2 or as specified using gl.set.verbosity]

Details

Filtering on Read Depth using the companion script gl.filter.rdepth can be on the basis of loci with exceptionally low counts, or loci with exceptionally high counts.

The minimum, maximum and mean read depth are provided. Output also is a histogram of read depth, accompanied by a box and whisker plot presented either in standard (boxplot="standard") or adjusted for skewness (boxplot=adjusted).

Refer to Tukey (1977, Exploratory Data Analysis. Addison-Wesley) for standard Box and Whisker Plots and Hubert & Vandervieren (2008), An Adjusted Boxplot for Skewed Distributions, Computational Statistics & Data Analysis 52:5186-5201) for adjusted Box and Whisker Plots.

Value

– dataframe with loci that are outliers

Author(s)

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

Examples

```
# SNP data
out <- gl.report.rdepth(testset.gl)
```

---

gl.report.RepAvg *Report summary of RepAvg averaged over both alleles for each locus or reproducibility (repeatability of the scores for fragment presence/absence).*

Description

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

Usage

```
gl.report.RepAvg(x, boxplot = "adjusted", range = 1.5, verbose = NULL)
```

Arguments

x	– name of the genlight object containing the SNP data [required]
boxplot	– if 'standard', plots a standard box and whisker plot; if 'adjusted', plots a boxplot adjusted for skewed distributions [default 'adjusted']
range	– specifies the range for delimiting outliers [default = 1.5 interquartile ranges]
verbose	– verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log ; 3, progress and results summary; 5, full report [default 2 or as specified using gl.set.verbosity]

Details

In the case of fragment presence/absence data (SilicoDArT), repeatability is the percentage of scores that are repeated in the technical replicate dataset.

A histogram and whisker plot are produced to aid in selecting a threshold.

Value

– Tabulation of repeatability against prospective Thresholds

Author(s)

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

Examples

```
# SNP data
  out <- gl.report.reproducibility(testset.gl)
# Tag P/A data
  out <- gl.report.reproducibility(testset.gs)
```

---

gl.report.reproducibility

Report summary of RepAvg (repeatability averaged over both alleles for each locus) or reproducibility (repeatability of the scores for fragment presence/absence).

Description

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

Usage

```
gl.report.reproducibility(x, boxplot = "adjusted", range = 1.5, verbose = NULL)
```

Arguments

x	– name of the genlight object containing the SNP data [required]
boxplot	– if 'standard', plots a standard box and whisker plot; if 'adjusted', plots a boxplot adjusted for skewed distributions [default 'adjusted']
range	– specifies the range for delimiting outliers [default = 1.5 interquartile ranges]
verbose	– verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log ; 3, progress and results summary; 5, full report [default 2 or as specified using gl.set.verbosity]

Details

In the case of fragment presence/absence data (SilicoDArT), repeatability is the percentage of scores that are repeated in the technical replicate dataset.

A histogram and whisker plot are produced to aid in selecting a threshold.

Value

– Tabulation of repeatability against prospective Thresholds

Author(s)

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

Examples

```
# SNP data
  out <- gl.report.reproducibility(testset.gl)
# Tag P/A data
  out <- gl.report.reproducibility(testset.gs)
```

---

gl.report.secondaries *Report loci containing secondary SNPs in sequence tags*

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.

Usage

```
gl.report.secondaries(x, boxplot = "adjusted", range = 1.5, verbose = 2)
```

Arguments

x	– name of the genlight object containing the SNP data [required]
boxplot	– if 'standard', plots a standard box and whisker plot; if 'adjusted', plots a boxplot adjusted for skewed distributions [default 'adjusted']
range	– specifies the range for delimiting outliers [default = 1.5 interquartile ranges]
verbose	– verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log ; 3, progress and results summary; 5, full report [default 2 or as specified using gl.set.verbosity]

Details

The script reports statistics associated with secondaries, and the consequences of filtering them out, and provides three plots. The first is a Box and Whisker plot adjusted to account for skewness, the second is a bargraph of the frequency of secondaries per sequence tag, and the third is Poisson expectation for those frequencies including an estimate of the zero class (no. of sequence tags with no SNP scored).

Heterozygosity in gl.report.heterozygosity is in a sense relative, because it is calculated against a background of only those loci that are polymorphic somewhere in the dataset. To allow intercomparability across studies and species, any measure of heterozygosity needs to accommodate loci that are invariant. However, the number of invariant loci are unknown given the SNPs are detected as single point mutational variants and invariant seqeunces are discarded, and because of the particular additional filtering pre-analysis. Modelling the counts of SNPs per sequence tag as a Poisson distribution in this script allows estimate of the zero class, that is, the number of invariant loci. This is reported, and the veracity of the estimate can be assessed by the correspondence of the observed frequencies against those under Poisson expectation in the associated graphs. The number of invariant loci can then be optionally provided to gl.report.heterozygosity via the parameter n.invariants.

Value

returns a genlight object of loci with multiple SNP calls

Author(s)

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

Examples

```
out <- gl.report.secondaries(bandicoot.gl)
```

---

gl.report.sexlinked *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.report.sexlinked(
  x,
  sex = NULL,
  t.het = 0,
  t.hom = 0,
  t.pres = 0,
```

```
  plot = TRUE,
  verbose = NULL
)
```

Arguments

x	– name of the genlight object containing the SNP data [required]
sex	– factor that defines the sex of individuals. See explanation above.
t.het	– tolerance, that is t.het=0.05 means that 5% of the heterogametic sex can be homozygous and still be regarded as consistent with a sex specific marker [default 0]
t.hom	– tolerance, that is t.hom=0.05 means that 5% of the homogametic sex can be heterozygous and still be regarded as consistent with a sex specific marker [default 0]
t.pres	– tolerance, that is t.pres=0.05 means that a silicodart marker can be present in either of the sexes and still be regarded as a sex-linked marker. [default 0]
plot	– creates a plot that shows the heterozygosity of males and females at each loci for SNP data or percentage of present/absent in the case of silicodart data. be regarded as consistent with a sex specific marker [default 0]
verbose	– 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 can be provided via a factor (equal to the length of the number of individuals) or to be held in the variable `x@other$ind.metrics$sex`. Coding is: M for male, F for female, U or NA for unknown/missing. 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

two list of sex specific loci, for XX/XY and ZZ/ZW systems.

Author(s)

Arthur Georges, Bernd Gruber & Floriaan Devloo-Delva (Post to https://groups.google.com/d/forum/dartr)

Examples

```
cat("does not work yet")
#result <- gl.sexlinkage(testset.gl)
```

gl.report.taglength *Report summary of sequence tag length across loci in a genlight adegenet object*

Description

SNP datasets generated by DArT typically have sequence tag lengths ranging from 20 to 69 base pairs.

Usage

```
gl.report.taglength(x, boxplot = "adjusted", range = 1.5, verbose = NULL)
```

Arguments

x	– name of the genlight object containing the SNP data [required]
boxplot	– if 'standard', plots a standard box and whisker plot; if 'adjusted', plots a boxplot adjusted for skewed distributions [default 'adjusted']
range	– specifies the range for delimiting outliers [default = 1.5 interquartile ranges]
verbose	– verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log ; 3, progress and results summary; 5, full report [default 2 or as specified using gl.set.verbosity]

Details

The minimum, maximum and mean of tag length are provided. Output also is a histogram of tag length, accompanied by a box and whisker plot presented either in standard (boxplot="standard") or adjusted for skewness (boxplot=adjusted).

Refer to Tukey (1977, Exploratory Data Analysis. Addison-Wesley) for standard Box and Whisker Plots and Hubert & Vandervieren (2008), An Adjusted Boxplot for Skewed Distributions, Computational Statistics & Data Analysis 52:5186-5201) for adjusted Box and Whisker Plots.

Value

– dataframe with loci that are outliers

Author(s)

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

Examples

```
out <- gl.report.taglength(testset.gl)
```

`gl.save` *Save an object in compressed binary format for later rapid retrieval.*

Description

This is a wrapper for saveRDS().

Usage

```
gl.save(x, file, verbose = NULL)
```

Arguments

`x`	– name of the genlight object containing SNP genotypes [required]
`file`	– name of the file to receive the binary version of the object [required]
`verbose`	– verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log ; 3, progress and results summary; 5, full report [default 2 or as specified using gl.set.verbosity]

Details

The script saves the object to the current workspace and returns the input gl object.

Value

the input object

Author(s)

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

Examples

```
gl.save(testset.gl,file.path(tempdir(),"testset.rds"))
gl.reloaded <- gl.load(file.path(tempdir(),"testset.rds"))
```

`gl.set.verbosity` *Set the default verbosity level*

Description

dartR functions have a verbosity parameter that sets the level of reporting during the execution of the function. The verbosity level, set by parameter 'verbose' can be one of verbose 0, silent or fatal errors; 1, begin and end; 2, progress log ; 3, progress and results summary; 5, full report. The default value for verbosity is stored as a flag in the gl@loc.metrics.flags slot of each genelight object. This script sets the value of the flag.

Usage

```
gl.set.verbosity(x, value = 2, verbose = NULL)
```

Arguments

`x`	name of the genlight object containing the SNP data, or the genind object containing the SilocoDArT data [required]
`value`	– set the default verbosity to be this value: 0, silent only fatal errors; 1, begin and end; 2, progress log ; 3, progress and results summary; 5, full report [default 2]
`verbose`	– 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 genlight with the verbosity flag reset

Author(s)

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

Examples

```
gl <- gl.set.verbosity(testset.gl, value=2)
```

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.stockR — *Generate a SNP matrix from a genlight {adegenet} object for subsequent use in R package stockR.*

Description

The script extracts the SNP data from the gl object as a matrix and transposes it to comply with the format expected by stockR.

Usage

```
gl.stockR(x, verbose = 2)
```

Arguments

x	– name of the genlight object containing SNP genotypes [required]
verbose	– 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 the transposed SNP matrix

Value

A matrix with the SNP scores in the form expected by stockR

Author(s)

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

Examples

```
gl <- gl.stockR(testset.gl)
```

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

Description

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

Usage

```
gl.subsample.loci(x, n, method = "random", mono.rm = FALSE, verbose = NULL)
```

Arguments

x	– name of the genlight object containing the SNP data [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']
mono.rm	– delete monomorphic loci before sampling [default FALSE]
verbose	– verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log ; 3, progress and results summary; 5, full report [default 2 or as specified using gl.set.verbosity]

Value

A genlight object with n loci

Author(s)

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

Examples

```
# SNP data
  gl2 <- gl.subsample.loci(testset.gl, n=200, method="pic")
# Tag P/A data
  gl2 <- gl.subsample.loci(testset.gl, n=100, method="random")
```

---

gl.test.heterozygosity

Tests the difference in heterozyosity between populations taken pairwise.

Description

Calculates the expected heterozygosities for each population in a genlight object, and uses re-randomization to test the statistical significance of differences in heterozyosity between populations taken pairwise.

Usage

```
gl.test.heterozygosity(
  x,
  nreps = 10000,
  alpha1 = 0.05,
  alpha2 = 0.01,
  plot = FALSE,
  plot.out = FALSE,
  verbose = NULL
)
```

Arguments

x	– a genlight object containing the SNP genotypes [Required]
nreps	– number of replications of the re-randomization [10,000]
alpha1	– significance level for comparison with diff=0 on plot [0.05]
alpha2	– second significance level for comparison with diff=0 on plot [0.01]
plot	– if TRUE, plots a sampling distribution of the differences for each comparison [FALSE]
plot.out	– if TRUE, outputs the plots as jpeg [FALSE]
verbose	– verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log ; 3, progress and results summary; 5, full report [default 2 or as specified using gl.set.verbosity]

Details

Optionally plots the sampling distribution for the difference between each pair of heterozygosities, marked with the critical limits alpha1 and alpha2, the observed heterozygosity, and the zero value (if in range). Can output the plots to jpeg.

Value

returns a dataframe containing population labels, heterozygosities and sample sizes

Author(s)

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

Examples

```
gl.report.heterozygosity(possums.gl[1:120,])
out <- gl.test.heterozygosity(possums.gl[1:120,], nreps=10, verbose=3, plot=TRUE)
```

---

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(
  x,
  type = "phylogram",
  outgroup = NULL,
  labelsize = 0.7,
  treefile = NULL,
  verbose = NULL
)
```

Arguments

x	– name of the genlight object containing the SNP 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]
treefile	– Name of the file for the tree topology using Newick format [Default NULL].
verbose	– specify the level of verbosity: 0, silent, fatal errors only; 1, flag function begin and end; 2, progress log ; 3, progress and results summary; 5, full report [default 2]

Value

A tree file of class phylo

Author(s)

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

Examples

```
# SNP data
  gl.tree.nj(testset.gl,type="fan")
# Tag P/A data
  gl.tree.nj(testset.gs,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.fixed.diff and 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,
  sympatric = FALSE,
  reps = 1000,
  delta = 0.02,
  verbose = NULL
)
```

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]
sympatric	– if TRUE, the two populations are sympatric, if FALSE then allopatric [FALSE]
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]
verbose	– 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 (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(x, outfile = "outfile.csv", outpath = tempdir(), verbose = NULL)
```

Arguments

x	– name of the genlight object containing the SNP data [required]
outfile	– file name of the output file (including extension) [default outfile.csv]
outpath	– path where to save the output file [default tempdir(), mandated by CRAN]. Use outpath=getwd() or outpath="." when calling this function to direct output files to your working directory.
verbose	– verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log ; 3, progress and results summary; 5, full report [default 2 or as specified using gl.set.verbosity]

Details

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

Value

saves a glenlight object to csv, returns NULL

Author(s)

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

Examples

```
# SNP data
  gl.write.csv(testset.gl, outfile="SNP_1row.csv")
# Tag P/A data
  gl.write.csv(testset.gs, outfile="PA_1row.csv")
```

gl2bayescan — *Convert a genlight object to format suitable for input to Bayescan*

Description

The output text file contains the snp data and relevant BAyescan command lines to guide input.

Usage

```
gl2bayescan(x, outfile = "bayescan.txt", outpath = tempdir(), verbose = NULL)
```

Arguments

x	– name of the genlight object containing the SNP data [required]
outfile	– file name of the output file (including extension) [default bayescan.txt]
outpath	– path where to save the output file [default tempdir(), mandated by CRAN]. Use outpath=getwd() or outpath="." when calling this function to direct output files to your working directory.
verbose	– verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log ; 3, progress and results summary; 5, full report [default 2 or as specified using gl.set.verbosity]

Author(s)

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

References

Foll M and OE Gaggiotti (2008) A genome scan method to identify selected loci appropriate for both dominant and codominant markers: A Bayesian perspective. Genetics 180: 977-993.

Examples

```
gl2bayescan(testset.gl)
```

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, verbose = NULL)
```

Arguments

gl	– name of the genlight object containing the SNP data [required]
verbose	– verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log ; 3, progress and results summary; 5, full report [default 2 or as specified using gl.set.verbosity]

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(
  x,
  method = 1,
  outfile = "output.fasta",
  outpath = tempdir(),
  probar = FALSE,
  verbose = NULL
)
```

Arguments

x	– 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)
probar	– if TRUE, a progress bar will be displayed for long loops [default = TRUE]

verbose – verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log ; 3, progress and results summary; 5, full report [default 2 or as specified using gl.set.verbosity]

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

Examples

```
gl <- gl.filter.reproducibility(testset.gl,t=1)
gl <- gl.filter.overshoot(gl,verbose=3)
gl <- gl.filter.callrate(testset.gl,t=.98)
gl <- gl.filter.monomorphs(gl)
gl2fasta(gl, method=1, outfile="test.fasta",verbose=3)
```

`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(
  x,
  outfile = "gl.str",
  outpath = tempdir(),
  probar = FALSE,
  verbose = NULL
)
```

Arguments

`x`	– name of the genlight object containing the SNP data [required]
`outfile`	– file name of the output file (including extension) [default gl.str]
`outpath`	– path where to save the output file [default tempdir(), mandated by CRAN]. Use outpath=getwd() or outpath="." when calling this function to direct output files to your working directory.
`probar`	switch to show/hide progress bar
`verbose`	– verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log ; 3, progress and results summary; 5, full report [default 2 or as specified using gl.set.verbosity]

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 gds format*

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(x, outfile = "gl2gds.gds", outpath = tempdir(), verbose = NULL)
```

Arguments

x	– name of the genlight object containing the SNP data [required]
outfile	– file name of the output file (including extension) [default gl2gds.gds]
outpath	– path where to save the output file [default tempdir(), mandated by CRAN]. Use outpath=getwd() or outpath="." when calling this function to direct output files to your working directory.
verbose	– verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log ; 3, progress and results summary; 5, full report [default 2 or as specified using gl.set.verbosity]

Author(s)

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

Examples

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

gl2genalex *Convert a genlight object to format suitable for input to genalex*

Description

The output csv file contains the snp data and other relevant lines suitable for genalex. This script is a wrapper for genind2genalex poppr

Usage

```
gl2genalex(x, outfile = "genalex.csv", outpath = tempdir(), verbose = NULL)
```

Arguments

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

Details

Reference: Peakall, R. and Smouse P.E. (2012) GenAlEx 6.5: genetic analysis in Excel. Population genetic software for teaching and research-an update. Bioinformatics 28, 2537-2539. http://bioinformatics.oxfordjournals.org

Author(s)

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

Examples

```
gl2genalex(testset.gl, outfile="testset.csv")
```

---

gl2gi *Converts a genlight object to genind object*

Description

Converts a genlight object to genind object

Usage

```
gl2gi(gl, probar = FALSE, verbose = NULL)
```

Arguments

gl	– a genlight object
probar	– if TRUE, a progress bar will be displayed for long loops [default = TRUE]
verbose	– verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log ; 3, progress and results summary; 5, full report [default 2 or as specified using gl.set.verbosity]

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

`gl2hiphop` *Converts genlight objects to hiphop format*

Description

This function exports genlight objects to the format used by the parentage assignment R package hiphop. Hiphop can be used for paternity and maternity assignment and outperforms conventional methods where closely related individuals occur in the pool of possible parents. The method compares the genotypes of offspring with any combination of potentials parents and scores the number of mismatches of these individuals at bi-allelic genetic markers (e.g. Single Nucleotide Polymorphisms).

Usage

```
gl2hiphop(gl, verbose = NULL)
```

Arguments

`gl`	– name of the genlight object containing the SNP data
`verbose`	– verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log ; 3, progress and results summary; 5, full report [default 2 or as specified using gl.set.verbosity]

Value

Dataframe containing all the genotyped individuals (offspring and potential parents) and their genotypes scored using bi-allelic markers.

Author(s)

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

References

Cockburn, A., Penalba, J.V.,Jaccoud, D.,Kilian, A., Brouwer, L.,Double, M.C., Margraf, N., Osmond, H.L., van de Pol, M. and Kruuk, L.E.B.(in revision). HIPHOP: improved paternity assignment among close relatives using a simple exclusion method for bi-allelic markers. Molecular Ecology Resources, DOI to be added upon acceptance

Examples

```
result <- gl2hiphop(testset.gl)
```

`gl2phylip` *Create a Phylip input distance matrix from a genlight (SNP) {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(
  x,
  outfile = "phyinput.txt",
  outpath = tempdir(),
  bstrap = 1,
  verbose = NULL
)
```

Arguments

`x`	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 [default tempdir(), mandated by CRAN]. Use outpath=getwd() or outpath="." when calling this function to direct output files to your working directory.
`bstrap`	Number of bootstrap replicates [default 1]
`verbose`	– verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log ; 3, progress and results summary; 5, full report [default 2 or as specified using gl.set.verbosity]

Value

Matrix of Euclidean distances between populations

Author(s)

Arthur Georges (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(), verbose = NULL)
```

Arguments

`x`	– name of the genlight object containing the SNP data [required]
`outfile`	– file name of the output file (including extension) [default plink.csv]
`outpath`	– path where to save the output file [default tempdir(), mandated by CRAN]. Use outpath=getwd() when calling this function or set.tempdir <- getwd() elsewhere in your script to direct output files to your working directory.
`verbose`	– verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log ; 3, progress and results summary; 5, full report [default 2 or as specified using gl.set.verbosity]

Author(s)

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

References

Purcell S, Neale B, Todd-Brown K, Thomas L, Ferreira MAR, Bender D, Maller J, Sklar P, de Bakker PIW, Daly MJ & Sham PC (2007). PLINK: a toolset for whole-genome association and population-based linkage analysis. American Journal of Human Genetics 81:551-575.

Examples

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

gl2related *Convert a genlight object to format suitable to be run with Coancestry*

Description

The output txt file contains the snp data and an additional column with the names if the individual. The file then can be used and loaded into coancestry or - if installed - run with the related package. Be aware the related package was crashing in previous versions, but in general is using the same code as coancestry and therefore should have identical results. Also running coancestry with thousands of SNPs via the GUI seems to be not reliable and therefore for comparisons between coancestry and related we suggest to use the command line version of coancestry.

Usage

```
gl2related(x, outfile = "related.txt", outpath = tempdir(), save = TRUE)
```

Arguments

x	– name of the genlight object containing the SNP data [required]
outfile	– file name of the output file (including extension) [default 'related.txt']
outpath	– path where to save the output file [default tempdir()]
save	– a switch if you want to save the file or not. This might be useful for someone who wants to use the coancestry function to calculate relatedness and not export to coancestry. See the example below. [default TRUE]

Value

a data.frame that can be used to run with the related package

Author(s)

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

References

Jack Pew, Jinliang Wang, Paul Muir and Tim Frasier (2014). related: related: an R package for analyzing pairwise relatedness data based on codominant molecular markers. R package version 0.8/r2. https://R-Forge.R-project.org/projects/related/

Examples

```
gtd <- gl2related(bandicoot.gl[1:10,1:20], save=FALSE)
## Not run:
##running with the related package
#install.packages("related", repos="http://R-Forge.R-project.org")
library(related)
coan <- coancestry(gtd, wang=1)
```

```
head(coan$relatedness)
##check ?coancestry for information how to use the function.

## End(Not run)
```

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

Description

This function exports a genlight object into a SNPassoc object. See package SNPassoc for details. #' This function needs package SNPassoc. At the time of writing (August 2020) the package was no longer available from CRAN. To install the package check their github repository. https://github.com/isglobal-brge/SNPassoc and/or use `install_github("isglobal-brge/SNPassoc")` to install the function and uncomment the function code.

Usage

```
gl2sa(x, verbose = NULL, installed = FALSE)
```

Arguments

`x`	– name of the genlight object containing the SNP data [required]
`verbose`	– verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log ; 3, progress and results summary; 5, full report [default 2 or as specified using gl.set.verbosity]
`installed`	switch to run the function once SNPassoc package is installed

Value

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

Author(s)

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

References

Gonzalez, J.R., Armengol, L., Sol?, X., Guin?, E., Mercader, J.M., Estivill, X. and Moreno, V. (2017). SNPassoc: an R package to perform whole genome association studies. Bioinformatics 23:654-655.

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 x@other$latlong and assumed to be in WGS84 coordinates, if not proj4 string is provided.

Usage

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

Arguments

x	– name of the genlight object containing the SNP data and location data, lat longs [required]
type	– type of output "kml" or "shp" [default 'shp']
proj4	– proj4string of data set (see spatialreference.org for projections) [default WGS84]
outfile	– name (path) of the output shape file [default 'gl']. shp extension is added automatically.
outpath	– path where to save the output file [default tempdir(), mandated by CRAN]. Use outpath=getwd() or outpath="." when calling this function to direct output files to your working directory.
verbose	– verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log ; 3, progress and results summary; 5, full report [default 2 or as specified using gl.set.verbosity]

Author(s)

Bernd Guber (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(), verbose = NULL)
```

Arguments

x	– name of the genlight object containing the SNP data [required]
outfile	– file name of the output file (including extension)[default snapp.nex]
outpath	– path where to save the output file [default tempdir(), mandated by CRAN]. Use outpath=getwd() or outpath="." when calling this function to direct output files to your working directory.
verbose	– verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log ; 3, progress and results summary; 5, full report [default 2 or as specified using gl.set.verbosity]

Author(s)

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

References

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.

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 gl2faststructure 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: doi.org/10.5281/zenodo.846816.

Usage

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

Arguments

x	– name of the genlight object containing the SNP data and location data, lat longs [required]
indNames	– specify individuals names to be added [if NULL, defaults to indNames(x)]
addcolumns	– additional columns to be added before genotypes [default NULL]
ploidy	– set the ploidy [defaults 2]
exportMarkerNames	– if TRUE, locus names locNames(x) will be included [default TRUE]
outfile	– file name of the output file (including extension) [default gl.str]
outpath	– path where to save the output file [default tempdir(), mandated by CRAN]. Use outpath=getwd() or outpath="." when calling this function to direct output files to your working directory.
verbose	– verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log ; 3, progress and results summary; 5, full report [default 2 or as specified using gl.set.verbosity]

Author(s)

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

Examples

```
#not run here
#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,
  verbose = NULL
)
```

Arguments

`x`	– name of the genlight object containing the SNP data or tag P/A data [required]
`outfile`	– file name of the output file (including extension) [default svd.nex]
`outpath`	– path where to save the output file [default tempdir(), mandated by CRAN]. Use outpath=getwd() when calling this function or set.tempdir <- getwd() elsewhere in your script to direct output files to your working directory.
`method`	– method = 1, nexus file with two lines per individual; method = 2, nexus file with one line per individual, ambiguity codes for SNP genotypes, 0 or 1 for presence/absence data [default 2]
`verbose`	– verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log ; 3, progress and results summary; 5, full report [default 2 or as specified using gl.set.verbosity]

Details

If the data are tag presence/absence, then method=2 is assumed.

Author(s)

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

References

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

Examples

```
gg <- testset.gl[1:20,1:100]
gg@other$loc.metrics <- gg@other$loc.metrics[1:100,]
gl2svdquartets(gg)
```

`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 will be gzipped before in order to be recognised by treemix. Plotting functions provided with treemix will need to be sourced from the treemix download page.

Usage

```
gl2treemix(
  x,
  outfile = "treemix_input.gz",
  outpath = tempdir(),
  verbose = NULL
)
```

Arguments

`x`	– name of the genlight object containing the SNP data [required]
`outfile`	– file name of the output file (including gz extension)[default treemix_input.gz]
`outpath`	– path where to save the output file [default tempdir(), mandated by CRAN]. Use outpath=getwd() when calling this function or set.tempdir <- getwd() elsewhere in your script to direct output files to your working directory.
`verbose`	– verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log ; 3, progress and results summary; 5, full report [default 2 or as specified using gl.set.verbosity]

Author(s)

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

References

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

Examples

```
gl2treemix(testset.gl, outpath=tempdir())
```

`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 or sequence tag frequency for the first population [required]
`s2`	– percentage SNP allele or sequence tag 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, parallel=FALSE)
df.loc <- data.frame(RepAvg = runif(nLoc(platy.gl)), CallRate = 1)
platy.gl@other$loc.metrics <- df.loc
gl.report.reproducibility(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

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

Description

This is a test data set on turtles. 250 individuals, 255 loci in >30 populations.

Usage

```
testset.gl
```

Format

genlight object

Author(s)

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

testset.gs *A genlight object created via the gl.read.silicodart function*

Description

This is a test data set on turtles. 218 individuals, 255 loci in >30 populations.

Usage

```
testset.gs
```

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 (glbugs@aerg.canberra.edu.au); 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.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

```
utils.dart2genlight(
  dart,
  ind.metafile = NULL,
  covfilename = NULL,
  probar = TRUE,
  verbose = 2
)
```

Arguments

dart	a dart object created via read.dart
ind.metafile	optional file in csv format with metadata for each individual (see details for explanation)
covfilename	depreciated, use parameter ind.metafile
probar	show progress bar
verbose	– 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 ind.metadata 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 metadata 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 ind.metadata file)

---

utils.dist.binary *Calculate a distance matrix for individuals defined in an {adegenet} genlight object using binary P/A data (SilicoDArT)*

Description

This script calculates various distances between individuals based on Tag Presence/Absence data. #'

The distance measure can be one of

simple – simple matching, both 1 or both 0 = 0; one 1 and the other 0 = 1. Presence and absence equally weighted. Jaccard – ignores matching 0, both 1 = 0; one 1 and the other 0 = 1. Absences could be for different reasons. Dice – both 0 = 0; both 1 = 2; one 1 and the other 0 = 1. Absences

could be for different reasons. Sometimes called the Czekanowski or Sorensen distance. Phi – binary analogue of the Pearson Correlation coefficient.

One might choose to disregard or downweight absences in comparison with presences because the homology of absences is less clear (mutation at one or the other, or both restriction sites). Your call.

Usage

```
utils.dist.binary(x, method = "simple", verbose = NULL)
```

Arguments

x	– name of the genlight containing the SNP genotypes [required]
method	– Specify distance measure [simple]
verbose	– verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log ; 3, progress and results summary; 5, full report [2]

Value

An object of class 'dist' giving distances between individuals

Author(s)

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

Examples

```
D <- utils.dist.binary(testset.gs, method="Jaccard")
```

---

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 (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 utils.prob.hwe() as developed by Wigginton, JE, Cutler, DJ, and Abecasis, GR.

Usage

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

Arguments

x	– a genlight object containting the SNP profiles for a population [Required]
prob	– level of significance [Default 0.05]
verbose	– verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log ; 3, progress and results summary; 5, full report [default 2]

Value

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

Author(s)

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

---

utils.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

```
utils.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

---

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

Description

Creates OutFLANK input file from individual genotype info.

Usage

```
utils.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.

`utils.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

```
utils.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.pa.ind *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

```
utils.pa.ind(x, unknown, nmin = 10, threshold = 0, verbose = NULL)
```

Arguments

x	– name of the input genlight object [required]
unknown	– 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]
verbose	– 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

returns 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').

Author(s)

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

Examples

```
# Test run with a focal individual from the Macleay River (EmmacMaclGeor)
#dartR:::utils.pa.ind(testset.gl, unknown="UC_00146", nmin=10, threshold=1, verbose=2)
```

`utils.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

```
utils.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 (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)
```

utils.read.dart *Import DarT data to R*

Description

Internal function called by gl.read.dart

Usage

```
utils.read.dart(
  filename,
  nas = "-",
  topskip = NULL,
  lastmetric = "RepAvg",
  verbose = 2
)
```

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

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, verbose = NULL)
```

Arguments

x	– name of the genlight object containing the SNP data [required]
verbose	– 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 OneRatioReflSnp, PICReflSnp 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)

See Also

utils.recalc.metrics for recalculating all metrics, utils.recalc.callrate for recalculating CallRate, utils.recalc.freqhomref for recalculating frequency of homozygous reference, utils.recalc.freqhomsnp for recalculating frequency of homozygous alternate, utils.recalc.freqhet for recalculating frequency of heterozygotes, gl.recalc.maf for recalculating minor allele frequency, gl.recalc.rdepth for recalculating average read depth

Examples

```
#out <- 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. It sets the Call Rate flag to TRUE.

Usage

```
utils.recalc.callrate(x, verbose = NULL)
```

Arguments

x	– name of the genlight object containing the SNP data [required]
verbose	– 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)

See Also

utils.recalc.metrics for recalculating all metrics, utils.recalc.avgpic for recalculating avg-PIC, utils.recalc.freqhomref for recalculating frequency of homozygous reference, utils.recalc.freqhomsnp for recalculating frequency of homozygous alternate, utils.recalc.freqhet for recalculating frequency of heterozygotes, gl.recalc.maf for recalculating minor allele frequency, gl.recalc.rdepth for recalculating average read depth

Examples

```
#out <- 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, verbose = NULL)
```

Arguments

x	– name of the genlight object containing the SNP data [required]
verbose	– 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)

See Also

utils.recalc.metrics for recalculating all metrics, utils.recalc.callrate for recalculating CallRate, utils.recalc.freqhomref for recalculating frequency of homozygous reference, utils.recalc.freqhomsnp for recalculating frequency of homozygous alternate, utils.recalc.AvgPIC for recalculating RepAvg, gl.recalc.maf for recalculating minor allele frequency, gl.recalc.rdepth for recalculating average read depth

Examples

```
#out <- 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, verbose = NULL)
```

Arguments

x	– name of the genlight object containing the SNP data [required]
verbose	– 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)

See Also

utils.recalc.metrics for recalculating all metrics, utils.recalc.callrate for recalculating CallRate, utils.recalc.avgpic for recalculating AvgPIC, utils.recalc.freqhomsnp for recalculating frequency of homozygous alternate, utils.recalc.freqhet for recalculating frequency of heterozygotes, gl.recalc.maf for recalculating minor allele frequency, gl.recalc.rdepth for recalculating average read depth

Examples

```
#result <- 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, verbose = NULL)
```

Arguments

x	– name of the genlight object containing the SNP data [required]
verbose	– 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)

See Also

utils.recalc.metrics for recalculating all metrics, utils.recalc.callrate for recalculating CallRate, utils.recalc.freqhomref for recalculating frequency of homozygous reference, utils.recalc.avgpic for recalculating AvgPIC, utils.recalc.freqhet for recalculating frequency of heterozygotes, gl.recalc.maf for recalculating minor allele frequency, gl.recalc.rdepth for recalculating average read depth

Examples

```
#out <- 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, verbose = NULL)
```

Arguments

x	– name of the genlight object containing the SNP data [required]
verbose	– 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)

See Also

utils.recalc.metrics for recalculating all metrics, utils.recalc.callrate for recalculating CallRate, utils.recalc.freqhomref for recalculating frequency of homozygous reference, utils.recalc.freqhomsnp for recalculating frequency of homozygous alternate, utils.recalc.freqhet for recalculating frequency of heterozygotes, gl.recalc.avgpic for recalculating AvgPIC, gl.recalc.rdepth for recalculating average read depth

Examples

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

---

utils.reset.flags *A utility script to reset to FALSE (or TRUE) the locus metric flags after some individuals or 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 applies also to some variable calculated by dartR (e.g. maf). This script resets the locus metrics flags to FALSE to indicate that these statistics in the genlight object are no longer current. The verbosity default is also set, and in the case of SilcoDArT, the flags PIC and OneRatio are also set.

Usage

```
utils.reset.flags(x, set = FALSE, value = 2, verbose = NULL)
```

Arguments

x	– name of the genlight object containing the SNP data or tag presence/absence data (SilicoDArT) [required]
set	– set the flags to TRUE or FALSE [FALSE]
value	– set the default verbosity for all functions, where verbosity is not specified
verbose	– verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log ; 3, progress and results summary; 5, full report [default NULL]

Details

If the locus metrics do not exist then they are added to the genlight object but not populated. If the locus metrics flags do not exist, then they are added to the genlight object and set to FALSE (or TRUE).

Value

The modified genlight object

Author(s)

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

See Also

utils.recalc.metrics for recalculating all metrics, utils.recalc.callrate for recalculating CallRate, utils.recalc.freqhomref for recalculating frequency of homozygous reference, utils.recalc.freqhomsnp for recalculating frequency of homozygous alternate, utils.recalc.freqhet for recalculating frequency of heterozygotes, gl.recalc.maf for recalculating minor allele frequency, gl.recalc.rdepth for recalculating average read depth

Examples

```
#result <- utils.reset.flags(testset.gl)
```

Index