

# Package ‘dartR’

December 6, 2017

**Type** Package

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

**Version** 1.0

**Date** 2017-12-06

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

**VignetteBuilder** knitr

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

**Imports** plyr, tidyr, reshape2, MASS, ggplot2, directlabels, quadprog, rgl, misc3d, pca3d, utils, Demerelate, seqinr, pegas, SNPassoc, methods, doParallel, stats, data.table, parallel, foreach, stringr, ape, vegan, SNPRelate, StAMPP, dismo, DBI, qvalue, plotly, sp, rgdal

**Suggests** knitr, rmarkdown, mmod, PopGenReport

**License** GPL-2

**LazyData** true

**RoxygenNote** 6.0.1

**NeedsCompilation** no

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

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

Repository CRAN

Date/Publication 2017-12-06 11:25:30 UTC

## R topics documented:

dart2genlight . . . . .	3
gi2gl . . . . .	4
gl.assign . . . . .	5
gl.collapse . . . . .	6
gl.collapse.recursive . . . . .	7
gl.dist . . . . .	8
gl.edit.recode.ind . . . . .	9
gl.edit.recode.pop . . . . .	10
gl.filter.callrate . . . . .	11
gl.filter.cloneid . . . . .	12
gl.filter.hamming . . . . .	13
gl.filter.hwe . . . . .	14
gl.filter.monomorphs . . . . .	15
gl.filter.repavg . . . . .	16
gl.filter.secondaries . . . . .	16
gl.fixed.diff . . . . .	17
gl.gene.freq . . . . .	18
gl.hwe.pop . . . . .	19
gl.ibd . . . . .	20
gl.make.recode.ind . . . . .	21
gl.make.recode.pop . . . . .	22
gl.outflank . . . . .	23
gl.pcoa . . . . .	24
gl.pcoa.plot . . . . .	25
gl.pcoa.plot.3d . . . . .	26
gl.pcoa.pop . . . . .	28
gl.pcoa.scree . . . . .	29
gl.percent.freq . . . . .	30
gl.read.dart . . . . .	30
gl.read.silicodart . . . . .	31
gl.recode.ind . . . . .	33
gl.recode.pop . . . . .	34
gl.report.bases . . . . .	35
gl.report.callrate . . . . .	36
gl.report.hamming . . . . .	36
gl.report.heterozygosity . . . . .	37
gl.report.hwe . . . . .	38
gl.report.ld . . . . .	39
gl.report.maf . . . . .	40
gl.report.monomorphs . . . . .	41
gl.report.pa . . . . .	41
gl.report.repavg . . . . .	42

gl.report.secondaries . . . . .	43
gl.subsample.loci . . . . .	44
gl.tree.nj . . . . .	44
gl.write.csv . . . . .	45
gl2demerelate . . . . .	46
gl2fasta . . . . .	47
gl2faststructure . . . . .	48
gl2gds . . . . .	49
gl2gi . . . . .	49
gl2nhyb . . . . .	50
gl2phylip . . . . .	51
gl2shp . . . . .	52
gl2structure . . . . .	53
is.fixed . . . . .	54
platy . . . . .	55
prob.hwe . . . . .	55
read.dart . . . . .	56
testset.gl . . . . .	57
testset_metadata . . . . .	57
testset_pop_recode . . . . .	58
testset_SNPs_2Row . . . . .	58
util.outflank . . . . .	59
util.outflank.MakeDiploidFSTMat . . . . .	60
util.outflank.plotter . . . . .	61
utils.hamming . . . . .	62
utils.hwe . . . . .	63

**Index****64**


---

dart2genlight	<i>Convert DarT to genlight</i>
---------------	---------------------------------

---

**Description**

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

**Usage**

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

**Arguments**

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

**Details**

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

**Value**

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

**Examples**

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

## End(Not run)
```

---

<code>gi2gl</code>	<i>Converts a <code>genind</code> object to <code>genlight</code> object</i>
--------------------	--

---

**Description**

Converts a `genind` object to `genlight` object

**Usage**

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

**Arguments**

<code>gi</code>	– a <code>genind</code> object
<code>parallel</code>	– switch to deactivate parallel version. Default set to <code>TRUE</code> . Only for testing purpose.

**Details**

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

**Value**

A `genlight` object, with all slots filled.

**Author(s)**

Bernd Gruber ([gbugs@aerg.canberra.edu.au](mailto:gbugs@aerg.canberra.edu.au))

---

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 ambiguous, on the basis of a weighted likelihood index.

### Usage

```
gl.assign(gl, id, nmin = 10, dim = NULL, alpha = 0.99, t = 0)
```

### Arguments

gl	– name of the input genlight object [required]
id	– identity label of the focal individual whose provenance is unknown [required]
nmin	– minimum sample size for a target population to be included in the analysis [default 10]
dim	– number of dimensions to retain in the dimension reduction [default k, number of populations]
alpha	– probability level for bounding ellipses in the PCoA plot [default 0.99]
t	– populations to retain for consideration; those for which the focal individual has less than or equal to t loci with private alleles [default 0]

### 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 tolerable 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  $\geq 10$ ).

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 threshold t).

### Author(s)

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

### Examples

```
x <- testset.gl
# Test run with a focal individual from the Macleay River (EmmacMaclGeor)
x <- gl.assign(testset.gl, id="UC_00146", nmin=10, alpha=0.95, t=1)
```

---

gl.collapse

*Collapse a distance matrix by amalgamating populations*

---

### Description

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

### Usage

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

### Arguments

fd	– name of the distance matrix produced by gl.fixed.diff() [required]
gl	– name of the genlight object from which the distance matrix was calculated [required]
recode.table	– name of the new recode.table to receive the new population reassignments arising from the amalgamation of populations [default tmp.csv]
tpop	– max number of fixed differences used amalgamating populations [default 0]
v	– verbosity = 0, silent; 1, brief; 2, verbose [default 1]

**Value**

The new genlight object with recoded populations

**Author(s)**

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

**Examples**

```
## Not run:
#only used the first 20 individuals due to runtime reasons
fd <- gl.fixed.diff(testset.gl[1:20,], tloc=0.05)
gl <- gl.collapse(fd, testset.gl, recode.table="testset_recode.csv", tpop=1)

## End(Not run)
```

---

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

---

**Description**

This script generates a fixed difference matrix from a genlight object {adegenet} and from it generates a population recode table used to amalgamate populations with a fixed difference count less than or equal to a specified tpop. The script then repeats the process until there is no further amalgamation of populations.

**Usage**

```
gl.collapse.recursive(gl, prefix = "collapse", tloc = 0, tpop = 2,
  nlimit = 10, v = 1)
```

**Arguments**

gl	– name of the genlight object from which the distance matrices are to be calculated [required]
prefix	– a string to be used as a prefix in generating the matrices of fixed differences (stored to disk) and the recode tables (also stored to disk) [default "collapse"]
tloc	– threshold defining a fixed difference (e.g. 0.05 implies 95:5 vs 5:95 is fixed) [default 0]
tpop	– max number of fixed differences allowed in amalgamating populations [default 0]
nlimit	– number of individuals combined, taking into account missing data, required for a pair of populations to be examined for fixed differences [default 2]
v	– verbosity = 0, silent; 1, brief; 2, verbose [default 1]

**Details**

The distance matrices are generated by `gl.fixed.diff()`, a recode table is generated using `gl.collapse()` and the resultant recode table is applied to the `genlight` object using `gl.recode.pop()`. The process is repeated as many times as necessary to yield a final table with no fixed differences less than or equal to the specified threshold, `tpop`. The intermediate and final recode tables and distance matrices are stored to disk as `csv` files for use with other analyses. In particular, the recode tables can be edited to replace population labels with meaningful names and reapplied in sequence.

**Value**

A new `genlight` object with recoded populations after amalgamations are complete.

**Author(s)**

Arthur Georges ([gbugs@aerg.canberra.edu.au](mailto:gbugs@aerg.canberra.edu.au))

**Examples**

```
#only used the first 20 individuals due to runtime reasons
fd <- gl.collapse.recursive(testset.gl[1:20,], prefix="testset", tloc=0, tpop=2)
```

---

<code>gl.dist</code>	<i>Calculate a distance matrix for populations defined in an {adegenet} genlight object</i>
----------------------	---

---

**Description**

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

**Usage**

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

**Arguments**

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



**Details**

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

**Value**

A matrix of distances between populations (class dist)

**Author(s)**

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

**Examples**

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

---

gl.edit.recode.ind	<i>Create or edit a individual (=specimen) names and create an re-code_ind file</i>
--------------------	---

---

**Description**

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

**Usage**

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

**Arguments**

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

**Details**

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

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

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

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

### Value

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

### Author(s)

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

### Examples

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

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

---

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

---

### Description

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

### Usage

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

### Arguments

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

### Details

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

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

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

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

### Value

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

### Author(s)

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

### Examples

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

## End(Not run)
```

---

gl.filter.callrate	<i>Filter loci or specimens in a genlight {adegenet} object based on call rate</i>
--------------------	--

---

### Description

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

### Usage

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

### Arguments

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

**Value**

The reduced genlight or genind object, plus a summary

**Author(s)**

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

**Examples**

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

---

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

---

**Description**

Filter for CloneID to select only unique SNPs

**Usage**

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

**Arguments**

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

**Value**

filtered genlight object, with unique cloneIDs

**Examples**

```
{  
}
```

---

gl.filter.hamming	<i>Filters loci in a genlight object based on pairwise Hamming distance between sequence tags</i>
-------------------	---

---

### Description

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

### Usage

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

### Arguments

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

### Details

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

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

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

Only one of two loci are retained if their Hamming distance is less than 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 ([gbugs@aerg.canberra.edu.au](mailto:gbugs@aerg.canberra.edu.au))

### Examples

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

---

gl.filter.hwe	<i>Filters loci that show significant departure from Hardy-Weinberg Equilibrium</i>
---------------	---

---

### Description

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

### Usage

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

### Arguments

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

### Details

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

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

### Value

a genlight object with the loci departing significantly from HWE removed

### Author(s)

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

### Examples

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

---

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

---

## Description

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

## Usage

```
gl.filter.monomorphs(gl, v = 1)
```

## Arguments

gl                   – name of the input genlight object [required]  
v                    – verbosity: 0, silent; 1, brief, 2; verbose if TRUE, silent if FALSE [default 1]

## Details

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

## Value

A genlight object with monomorphic loci removed

## Author(s)

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

## Examples

```
## Not run:  
gl <- gl.filter.monomorphs(gl)  
  
## End(Not run)
```

---

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

---

### Description

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

### Usage

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

### Arguments

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

### Value

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

### Author(s)

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

### Examples

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

---

gl.filter.secondaries      *Filter loci that represent secondary SNPs in a genlight {adegenet} object*

---

### Description

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

### Usage

```
gl.filter.secondaries(gl)
```



**Arguments**

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

**Value**

The reduced genlight, plus a summary

**Author(s)**

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

**Examples**

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

---

gl.fixed.diff	<i>Generate a matrix of fixed differences from a genlight or genind object {adegenet}</i>
---------------	---

---

**Description**

This script takes SNP data grouped into populations in a genlight object (DARtSeq) and generates a matrix of fixed differences between populations taken pairwise

**Usage**

```
gl.fixed.diff(gl, tloc = 0, nlimit = 2, pc = FALSE, v = 1)
```

**Arguments**

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

tloc               – threshold defining a fixed difference (e.g. 0.05 implies 95:5 vs 5:95 is fixed) [default 0]

nlimit           – number of individuals with non-missing SNP scores in the two populations combined, required for scoring of fixed differences [default 2]

pc               – logical value indicating whether to report fixed difference counts (pc=FALSE) or percentages (pc=TRUE) [default TRUE]

v               – verbosity = 0, silent; 1, brief; 2, verbose [default 1]

**Details**

A fixed difference at a locus occurs when two populations share no alleles. The challenge with this approach is that when sample sizes are finite, fixed differences will occur through sampling error, compounded when many loci are examined. Simulations suggest that sample sizes of  $n_1=5$  and  $n_2=5$  is adequate to reduce the probability of [experiment-wide] type 1 error to negligible levels [ploidy=2]. A warning is issued if comparison between two populations involves sample sizes less than 5, taking into account allele drop-out. The minimum sample size for scoring fixed differences between two populations can be set with the parameter `nlimit`. An absolute fixed difference is as defined above. However, one might wish to score fixed differences at some lower level of allele frequency difference, say where percent allele frequencies 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**

Matrix of percent fixed differences (lower matrix), number of loci (upper matrix)

**Author(s)**

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

**See Also**

[is.fixed](#)

**Examples**

```
#only used the first 20 individuals due to runtime reasons
mat <- gl.fixed.diff(testset.gl[1:20,], tloc=0.05, nlimit=5)
```

---

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 (glbugs@aerg.canberra.edu.au)

**Examples**

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

---

gl.hwe.pop

*Filter function to facilitate analysing of dart data*

---

**Description**

Filter function to facilitate analysing of dart data

**Usage**

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

**Arguments**

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

**Value**

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

**Examples**

```
## Not run:
gl.hwe.pop(gi=gi, pvalue = 0.05, plot = TRUE)

## End(Not run)
```

---

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`), `stampFst` (package `StAMPP`) and `Mercator` (package `dismo`).

**Usage**

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

**Arguments**

gl	genlight object. If provided a standard analysis on Fst/1-Fst and log(distance) is performed
Dgen	genetic distance matrix if no genlight object with coordinates is provided
Dgeo	Euclidean distance matrix if no genlight object is provided
projected	Switch to indicate that coordinates are already projected (not in lat long) and therefore no projection is carried out. Default is FALSE, so it is assumed coordinates are in lat/longs.
permutations	number of permutations in the mantel test
plot	should an isolation by distance plot be returned. Default is plot=TRUE

**Value**

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

**Author(s)**

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

**References**

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

**See Also**

[mantel](#), [stampFst](#), [Mercator](#)

**Examples**

```
## Not run:
gl <- gl.ibd(gl=testset.gl)

## End(Not run)
```

---

gl.make.recode.ind	<i>Create a proforma recode_ind file for reassigning individual (=specimen) names</i>
--------------------	---

---

**Description**

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

**Usage**

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

**Arguments**

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

**Details**

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

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

**Value**

A vector containing the new individual names

**Author(s)**

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

**Examples**

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

## End(Not run)
```

---

gl.make.recode.pop	<i>Create a proforma recode_ind_table file for reassigning individual (=specimen) names</i>
--------------------	---

---

**Description**

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

**Usage**

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

**Arguments**

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

**Details**

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

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

**Value**

A vector containing the new individual names

**Author(s)**

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

**Examples**

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

## End(Not run)
```

---

gl.outflank	<i>Function to identify loci under selection per population using the outflank method of Whitlock and Lotterhos (2015)</i>
-------------	--

---

**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 function is applied.
RightTrimFraction	The proportion of loci that are trimmed from the upper end of the range of Fst before the likelihood function 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  $F_{st}$ . *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**

```
## Not run:
gl.outflank(gi, plot = TRUE)

## End(Not run)
```

---

gl.pcoa

*PCoA ordination (glPca)*

---

**Description**

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

**Usage**

```
gl.pcoa(gl, nfactors = 5, v = TRUE, parallel = FALSE)
```

**Arguments**

gl	Name of the genlight object containing the SNP genotypes by specimen and population [required]
nfactors	Number of dimensions to retain in the output file [default 5]
v	– verbose if TRUE, silent if FALSE [default TRUE]
parallel	should parallel version be used (does fail under Windows)



**Value**

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

**Author(s)**

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

**Examples**

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

---

gl.pcoa.plot

*Bivariate plot of the results of a PCoA ordination*

---

**Description**

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

**Usage**

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

**Arguments**

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

## Details

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

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

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

## Value

A plot of the ordination

## Author(s)

Arthur Georges ([glbugs@aerg.canberra.edu.au](mailto:glbugs@aerg.canberra.edu.au))

## Examples

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

---

`gl.pcoa.plot.3d`

*3D interactive plot of the results of a PCoA ordination*

---

## Description

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

## Usage

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

**Arguments**

x	– name of the glPca object containing the factor scores and eigenvalues [required]
gl	– name of the genlight object from which the PCoA was generated
title	– a title for the plot [default "PCoA"]
xaxis	– identify the x axis from those available in the ordination (xaxis <= nfactors) [default 1]
yaxis	– identify the y axis from those available in the ordination (yaxis <= nfactors) [default 2]
zaxis	– identify the z axis from those available in the ordination (zaxis <= nfactors) [default 3]
shape	– shape of the points, one of sphere, tetraedron or cube [default "sphere"]
radius	– size of the points [default 2]
legend	– one of bottomright, bottom, bottomleft, left, topleft, top, topright, right, center [default "bottom"]

**Details**

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

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

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

**Value**

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

**Author(s)**

Arthur Georges ([glbugs@aerg.canberra.edu.au](mailto:glbugs@aerg.canberra.edu.au))

**Examples**

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

---

`gl.pcoa.pop`*PCoA ordination of populations*

---

## Description

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

## Usage

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

## Arguments

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

## Value

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

## Author(s)

Arthur Georges ([gl.bugs@aerg.canberra.edu.au](mailto:gl.bugs@aerg.canberra.edu.au))

## Examples

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

---

gl.pcoa.scree	<i>Produce a plot of eigenvalues, standardized as percentages, derived from a PCoA</i>
---------------	--

---

### Description

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

### Usage

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

### Arguments

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

### Details

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

### Value

The scree plot

### Author(s)

Arthur Georges ([glbugs@aerg.canberra.edu.au](mailto:glbugs@aerg.canberra.edu.au))

### Examples

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

---

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

---

### Description

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

### Usage

```
gl.percent.freq(gl, v = 1)
```

### Arguments

gl                    – name of the genlight containing the SNP genotypes or genind object containing the presence/absence data [required]  
v                     – verbosity = 0, silent; 1, brief; 2, verbose [default 1]

### Value

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

### Author(s)

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

### Examples

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

---

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

---

### Description

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

### Usage

```
gl.read.dart(filename, covfilename = NULL, nas = "-", topskip = NULL,
lastmetric = "RepAvg", probar = TRUE)
```

**Arguments**

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

**Value**

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

**Examples**

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

---

gl.read.silicodart	<i>Import presence/absence data from SilicoDArT and convert to genind {agegenet} format</i>
--------------------	---

---

**Description**

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

**Usage**

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

**Arguments**

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

**Details**

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

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

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

**Value**

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

**Author(s)**

Arthur Georges ([gbugs@aerg.canberra.edu.au](mailto:gbugs@aerg.canberra.edu.au))

**Examples**

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

## End(Not run)
```



---

gl.recode.ind	<i>Recode individual (=specimen) labels in a genlight or genind object {adegenet}</i>
---------------	---

---

### Description

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

### Usage

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

### Arguments

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

### Details

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

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

### Value

A genlight or genind object with the recoded and reduced data

### Author(s)

Arthur Georges ([glbugs@aerg.canberra.edu.au](mailto:glbugs@aerg.canberra.edu.au))

### See Also

[gl.filter.monomorphs](#)

### Examples

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

---

gl.recode.pop	<i>Recode population assignments in a genelight or genind object {adegenet}</i>
---------------	---

---

## Description

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

## Usage

```
gl.recode.pop(gl, pop.recode, v = 1)
```

## Arguments

gl	– name of the genelight object containing SNP genotypes or a genind object containing presence/absence data [required]
pop.recode	– name of the csv file containing the population reassignments [required]
v	– verbosity: 0, silent; 1, brief; 2, verbose [default 1]

## Details

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

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

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

## Value

A genelight or genind object with the recoded and reduced data

## Author(s)

Arthur Georges ([glbugs@aerg.canberra.edu.au](mailto:glbugs@aerg.canberra.edu.au))

## See Also

[gl.filter.monomorphs](#)

## Examples

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

## End(Not run)
```

---

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

---

## Description

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

## Usage

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

## Arguments

gl                    – name of the DArT genlight object [required]

## Details

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

## Value

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

## Author(s)

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

## Examples

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

---

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

---

### Description

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

### Usage

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

### Arguments

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

### Value

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

### Author(s)

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

### Examples

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

---

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

---

### Description

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

**Usage**

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

**Arguments**

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

**Details**

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

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

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

**Value**

Histogram of Hamming distance for the gl object

**Author(s)**

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

**Examples**

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

---

```
gl.report.heterozygosity  
Reports hetrozygosity
```

---

**Description**

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

**Usage**

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

**Arguments**

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

**Details**

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

**Value**

a dataframe containing population labels, observed heterozygosities and sample sizes

**Author(s)**

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

**Examples**

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

---

gl.report.hwe

*Reports departure from Hardy-Weinberg Equilibrium*

---

**Description**

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

**Usage**

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

**Arguments**

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

**Details**

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

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

**Value**

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

**Author(s)**

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

**Examples**

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

---

gl.report.ld	<i>Calculates pairwise population based Linkage Disequilibrium across all loci using the specified number of cores</i>
--------------	--

---

**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 may crash half way. Also remove loci with have only missing value before running the function.

**Usage**

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

**Arguments**

gi	a genlight or genind object created (genlight objects are internally converted via <a href="#">gl2gi</a> to genind)
name	character string for rdata file. If not given genind object name is used
save	switch if results are saved in a file
nchunks	how many subchunks will be used (the less the faster, but if the routine crashes more bits are lost)
ncores	how many cores should be used
chunkname	the name of the chunks for saving, default is NULL

**Value**

returns calculation of pairwise LD across all loci between subpopulation. This functions uses if specified many cores on your computer to speed up. And if save is used can restart (if save=TRUE is used) with the same command starting where it crashed. The final output is a data frame that holds all statistics of pairwise LD between loci. (See ?LD in package genetics for details).

**Author(s)**

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

---

gl.report.maf

*Report minor allele frequency for loci*

---

**Description**

Minor allele frequencies are reported for every loci. Also the number of individuals that have been used to calculate allele frequencies are reported to be able to discard frequencies of loci where only a few individuals are available.

**Usage**

```
gl.report.maf(gl, minind = 5, plot = TRUE)
```

**Arguments**

gl	– name of the genlight object containing the SNP data [required]
minind	– minimum number of individuals required to calculate the minor allele frequency. Please note allele frequencies for such loci are still reported, but a warning is issued if the number is below minind.
plot	– switch if a histogram of the distribution of allele frequencies should be provided

**Value**

the functions returns a list with two slots (vectors). maf contains the minor allele frequency and n the number of individuals the minor allele frequency is based on.

**Author(s)**

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

**Examples**

```
frequ <- gl.report.maf(testset.gl)
summary(frequ$maf)
summary(frequ$n)
```



---

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

---

**Description**

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

**Usage**

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

**Arguments**

gl                   – name of the input genlight object [required]

**Details**

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

**Value**

A report on loci, polymorphic, monomorphic, all NAs

**Author(s)**

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

**Examples**

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

---

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

---

**Description**

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

**Usage**

```
gl.report.pa(gl, id, nmin = 10, t = 0)
```

**Arguments**

- gl                   – name of the input genlight object [required]
- id                   – identity label of the focal individual whose provenance is unknown [required]
- nmin                – minimum sample size for a target population to be included in the analysis [default 10]
- t                    – populations to retain in the output genlight object; those for which the focal individual has less than or equal to t loci with private alleles [default 0]

**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 threshold t).

**Author(s)**

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

**Examples**

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

---

<code>gl.report.repavg</code>	<i>Report summary of RepAvg, reproducibility averaged over both alleles for each locus in a genlight adegenet object</i>
-------------------------------	--

---

**Description**

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

**Usage**

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

**Arguments**

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

**Value**

- the mean call rate

**Author(s)**

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

**Examples**

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

---

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

---

**Description**

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

**Usage**

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

**Arguments**

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

**Value**

1

**Author(s)**

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

**Examples**

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

---

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

---

### Description

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

### Usage

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

### Arguments

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

### Value

A genlight object with n loci

### Author(s)

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

### Examples

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

## End(Not run)
```

---

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

---

### Description

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

**Usage**

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

**Arguments**

gl                   – Name of the genlight object containing the SNP data or a genind object containing presence absence data [required]

type                 – Type of dendrogram phylogram/cladogram/fan/unrooted [Default Phylogram]

outgroup            – Vector containing the population names that are the outgroups [Default NULL]

labelsize           – Size of the labels as a proportion of the graphics default [Default 0.7]

**Value**

A tree file of type phylo

**Author(s)**

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

**Examples**

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

---

gl.write.csv

*Write out data from a gl object [adegenet](#) to csv file*

---

**Description**

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

**Usage**

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

**Arguments**

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

outfile             – name of the csv file to write the data to [required]

**Details**

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

**Value**

saves a genlight object to csv

**Author(s)**

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

**Examples**

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

---

gl2demerelate	<i>Create a dataframe suitable for input to package {Demerelate} from a genlight {adegenet} object</i>
---------------	--

---

**Description**

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

**Usage**

```
gl2demerelate(gl)
```

**Arguments**

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

**Value**

A dataframe suitable as input to package {Demerelate}

**Author(s)**

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

**Examples**

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

---

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

---

### Description

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

### Usage

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

### Arguments

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

### Details

Four methods are employed

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

Method=2 – the heterozygous 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 heterozygous 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 'Trimmed-Sequence' 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 (glbugs@aerg.canberra.edu.au)

**Examples**

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

---

gl2faststructure	<i>Export DArT genlight object {adegenet} to faststructure format (to run faststructure elsewhere)</i>
------------------	--

---

**Description**

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

**Usage**

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

**Arguments**

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

**Details**

The script writes out the a file in faststructure format.

**Author(s)**

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



---

gl2gds	<i>Convert a genlight object to gds format SNPRelate</i>
--------	--

---

**Description**

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

**Usage**

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

**Arguments**

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

**Author(s)**

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

**Examples**

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

---

gl2gi	<i>Converts a genlight object to genind object</i>
-------	--

---

**Description**

Converts a genlight object to genind object

**Usage**

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

**Arguments**

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

**Details**

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

**Value**

A genind object, with all slots filled.

**Author(s)**

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

---

gl2nhyb

*Create an input file for the program NewHybrids*

---

**Description**

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

**Usage**

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

**Arguments**

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

**Details**

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

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

**Value**

The reduced genlight object

**Author(s)**

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

**Examples**

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

## End(Not run)
```

---

gl2phylip	<i>Create a Phylip input distance matrix from a genlight (SNP) or genind (SilicoDarT) {adegenet} object</i>
-----------	---

---

**Description**

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

**Usage**

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

**Arguments**

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

**Value**

Matrix of Euclidean distances between populations

**Author(s)**

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

**Examples**

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

## End(Not run)
```

---

`gl2shp`*Convert genlight objects to ESRI shapefiles*

---

### Description

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

### Usage

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

### Arguments

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

### Author(s)

Bernd Guber ([glbugs@aerg.canberra.edu.au](mailto:glbugs@aerg.canberra.edu.au))

### Examples

```
## Not run:

gl2shp(testset.gl)

## End(Not run)
```

---

`gl2structure`*Converts genlight objects to STRUCTURE formatted 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](https://github.com/lvclark/R_genetics_conv)) and this function is basically a wrapper around his numeric2structure function.

## Usage

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

## Arguments

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

## Author(s)

Lindsay V. Clark [lvclark@illinois.edu], wrapper by Bernd Gruber

## Examples

```
## Not run:
gl2structure(testset.gl)

## End(Not run)
```

---

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

---

### Description

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

### Usage

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

### Arguments

<code>s1</code>	– percentage SNP allele frequency for the first population [required]
<code>s2</code>	– percentage SNP allele frequency for the second population [required]
<code>t</code>	– 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). Tolerance in the definition of a fixed difference is provided by the `t` parameter. For example, `t=0.05` means that SNP allele frequencies of 95,5 and 5,95 percent will be reported as fixed (TRUE).

### Value

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

### Author(s)

Arthur Georges ([glbugs@aerg.canberra.edu.au](mailto:glbugs@aerg.canberra.edu.au))

### See Also

[gl.fixed.diff](#)

### Examples

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

---

 platy

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


---

**Description**

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

**Format**

csv

**Author(s)**

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

**Examples**

```
## Not run:
library(PopGenReport)
read.csv( paste(.libPaths()[1],"/dartR/extdata/platy.csv",sep="" ))
platy <- read.genetable( paste(.libPaths()[1],"/dartR/extdata/platy.csv",
sep="" ), ind=1, pop=2, lat=3, long=4, other.min=5, other.max=6, oneColPerAll=FALSE,
sep="/")
platy.gl <- (gi2gl(platy))
df.loc <- data.frame(RepAvg = runif(nLoc(platy.gl)), CallRate = 1)
platy.gl@other$loc.metrics <- df.loc
gl.report.repavg(platy.gl)

## End(Not run)
```

---

 prob.hwe

*Exact SNP test of Hardy-Weinberg Equilibrium*


---

**Description**

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

**Usage**

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

**Arguments**

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

**Details**

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

**Value**

Exact probability of agreement with HWE

**Author(s)**

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

**Examples**

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

---

read.dart

*Import DarT data to R*

---

**Description**

Internal function called by gl.read.dart

**Usage**

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

**Arguments**

filename	path to file (csv file only currently)
nas	a character specifying NAs (default is "-")
topskip	a number specifying the number of rows to be skipped. If not provided the number of rows to be skipped are "guessed" by the number of rows with "*" at the beginning.
lastmetric	specifies the last non genetic column (Default is "RepAvg"). Be sure to check if that is true, otherwise the number of individuals will not match. You can also specify the last column by a number.

**Value**

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



---

testset.gl	<i>A genlight object created via the read.dart functions</i>
------------	--

---

**Description**

A genlight object created via the read.dart functions

**Usage**

testset.gl

**Format**

genlight object

**Author(s)**

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

---

testset_metadata	<i>Metadata file. Can be integrated via the dart2genlight function.</i>
------------------	---

---

**Description**

Metadata file. Can be integrated via the dart2genlight function.

**Format**

csv

**Author(s)**

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

---

testset\_pop\_recode      *Recode file to be used with the function.*

---

**Description**

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

**Format**

csv

**Author(s)**

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

---

testset\_SNPs\_2Row      *Testfile in DArT format (as provided by DArT)*

---

**Description**

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

**Format**

csv

**Author(s)**

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

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 convenient 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 function is applied.

RightTrimFraction	The proportion of loci that are trimmed from the upper end of the range of Fst before the likelihood function 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 significantly low FST (not reliable)
- numberHighFstOutliers: Number of loci identified as having 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 distribution (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 histogram of the FST

---

utils.hamming	<i>Calculates the Hamming distance between two DArT trimmed DNA sequences</i>
---------------	---

---

**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 (gbugs@aerg.canberra.edu.au)

### Examples

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

---

utils.hwe	<i>Calculates departure from Hardy-Weinberg Equilibrium. Utility script not meant for end users.</i>
-----------	--

---

### Description

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

### Usage

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

### Arguments

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

### Value

Locus, Hom\_1, Het, Hom\_2, N, Prob, Sig, BonSig)

### Author(s)

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

# Index

## \*Topic **datasets**

- platy, [55](#)
  - testset.gl, [57](#)
  - testset\_metadata, [57](#)
  - testset\_pop\_recode, [58](#)
  - testset\_SNPs\_2Row, [58](#)
- adegenet, [3](#), [45](#)
- dart2genlight, [3](#), [31](#)
- gi2gl, [4](#)
- gl.assign, [5](#)
- gl.collapse, [6](#)
- gl.collapse.recursive, [7](#)
- gl.dist, [8](#)
- gl.edit.recode.ind, [9](#)
- gl.edit.recode.pop, [10](#)
- gl.filter.callrate, [11](#)
- gl.filter.cloneid, [12](#)
- gl.filter.hamming, [13](#)
- gl.filter.hwe, [14](#)
- gl.filter.monomorphs, [15](#), [33](#), [34](#)
- gl.filter.repavg, [16](#)
- gl.filter.secondaries, [16](#)
- gl.fixed.diff, [17](#), [54](#)
- gl.gene.freq, [18](#)
- gl.hwe.pop, [19](#)
- gl.ibd, [20](#)
- gl.make.recode.ind, [21](#)
- gl.make.recode.pop, [22](#)
- gl.outflank, [23](#)
- gl.pcoa, [24](#)
- gl.pcoa.plot, [25](#)
- gl.pcoa.plot.3d, [26](#)
- gl.pcoa.pop, [28](#)
- gl.pcoa.scree, [29](#)
- gl.percent.freq, [30](#)
- gl.read.dart, [30](#)
- gl.read.silicodart, [31](#)
- gl.recode.ind, [33](#)
- gl.recode.pop, [34](#)
- gl.report.bases, [35](#)
- gl.report.callrate, [36](#)
- gl.report.hamming, [36](#)
- gl.report.heterozygosity, [37](#)
- gl.report.hwe, [38](#)
- gl.report.ld, [39](#)
- gl.report.maf, [40](#)
- gl.report.monomorphs, [41](#)
- gl.report.pa, [41](#)
- gl.report.repavg, [42](#)
- gl.report.secondaries, [43](#)
- gl.subsample.loci, [44](#)
- gl.tree.nj, [44](#)
- gl.write.csv, [45](#)
- gl2demerelate, [46](#)
- gl2fasta, [47](#)
- gl2faststructure, [48](#)
- gl2gds, [49](#)
- gl2gi, [39](#), [49](#)
- gl2nhyb, [50](#)
- gl2phylip, [51](#)
- gl2shp, [52](#)
- gl2structure, [53](#)
- is.fixed, [18](#), [54](#)
- mantel, [20](#), [21](#)
- Mercator, [20](#), [21](#)
- platy, [55](#)
- prob.hwe, [55](#)
- read.dart, [31](#), [56](#)
- stampFst, [20](#), [21](#)
- testset.gl, [57](#)
- testset\_metadata, [57](#)
- testset\_pop\_recode, [58](#)



testset\_SNPs\_2Row, [58](#)

util.outflank, [24](#), [59](#)

util.outflank.MakeDiploidFSTMat, [24](#), [60](#)

util.outflank.plotter, [24](#), [61](#)

utils.hamming, [62](#)

utils.hwe, [63](#)