Package 'dartR.popgen'

November 18, 2026

Type Package

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

Version 1.3.0

Date 2026-11-18

Revision Hardy Hedgehog

Description Facilitates the analysis of SNP (single nucleotide polymorphism) and silicodart (presence/absence) data. 'dartR.popgen' provides a suit of functions to analyse such data in a population genetics context. It provides several functions to calculate population genetic metrics and to study population structure. Quite a few functions need additional software to be able to run (gl.run.structure(), gl.blast(), gl.LDNe()). You find detailed description in the help pages how to download and link the packages so the function can run the software. 'dartR.popgen' is part of the the 'dartRverse' suit of packages. Gruber et al. (2018) <doi:10.1111/1755-0998.12745>. Mijangos et al. (2022) <doi:10.1111/2041-210X.13918>.

Encoding UTF-8

Depends R (>= 4.1.0), dartR.base (>= 1.3.0), dartR.data

Imports methods, adegenet (>= 2.0.0), utils, MASS, dplyr, patchwork, crayon, ggplot2, data.table, stringr, furrr, future, ggdendro, LEA, pillar, plyr, terra, purrr, R.utils, ape, StAMPP, stats, parallel

Suggests assignPOP, coda, directlabels, pegas, SIBER, expm, fields, gplots, gridExtra, igraph, iterpc, label.switching, leaflet, proxy, qvalue, raster, reshape2, scales, snpStats, tidyr, viridis, zoo, sp, testthat, withr

License GPL (>= 3)

RoxygenNote 7.3.3

NeedsCompilation no

Author Bernd Gruber [aut, cre], Arthur Georges [aut], Jose L. Mijangos [aut],

Carlo Pacioni [aut],
Diana Robledo-Ruiz [aut],
Peter J. Unmack [ctb],
Oliver Berry [ctb],
Lindsay V. Clark [ctb],
Floriaan Devloo-Delva [ctb],
Eric Archer [ctb],
Ching Ching Lau [ctb]

URL https://green-striped-gecko.github.io/dartR/

BugReports https://groups.google.com/g/dartr?pli=1

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

Repository CRAN

Date/Publication 2026-11-18 10:10:11 UTC

Contents

gl.amova . . . 3
gl.assign.mahalanobis . . . 4
gl.assign.on.genotype . . . 7
gl.assign.pa . . . 9
gl.assign.pca . . . 10
gl.blast . . . 12
gl.check.future.panel . . . 16
gl.check.panel . . . 19
gl.collapse . . . 23
gl.evanno . . . 24
gl.find.genes.for.loci . . . 26
gl.find.loci.in.genes . . . 28
gl.ld.distance . . . 30
gl.ld.haplotype . . . 32
gl.LDNe . . . 35
gl.map.popcluster . . . 37
gl.map.snmf . . . 39
gl.map.structure . . . 41
gl.nhybrids . . . 44
gl.opt.ind.all . . . 46
gl.outflank . . . 49
gl.plot.faststructure . . . 51
gl.plot.popcluster . . . 54
gl.plot.snmf . . . 56
gl.plot.structure . . . 58
gl.read.structure . . . 61
gl.run.assignpop . . . 63
gl.run.bayescan . . . 65
gl.run.epos . . . 69
gl.run.faststructure . . . 73

gl.run.mongrail 75
gl.run.popcluster 77
gl.run.snmf 80
gl.run.stairway2 82
gl.run.structure 86
gl.select.panel 90
gl.select.panel.combined 92
gl.select.panel.parallel 95
gl.sfs 97
gl.TajimasD 99
utils.get.allele.freq 101
utils.outflank 102
utils.outflank.MakeDiploidFSTMat 104
utils.outflank.plotter 105
utils.structure.evanno 106
utils.structure.genind2gtypes 106
utils.structure.run 107

Index **110**

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

Description

This script performs an AMOVA based on the genetic distance matrix from stamppNeisD() [package StAMPP] using the amova() function from the package PEGAS for exploring within and between population variation. For detailed information use their help pages: ?pegas::amova, ?StAMPP::stamppAmova. Be aware due to a conflict of the amova functions from various packages I had to 'hack' StAMPP::stamppAmova to avoid a namespace conflict.

Usage

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

Arguments

`x`	Name of the genlight containing the SNP genotypes, with population information (at least two populations) [required].
`distance`	Distance matrix (class matrix or dist) between individuals, labelled with the individual names (if not provided NeisD from StAMPP::stamppNeisD is calculated) [default NULL].
`permutations`	Number of permutations to perform for hypothesis testing [default 100]. Please note should be set to 1000 for analysis.
`verbose`	Verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log; 3, progress and results summary; 5, full report [default NULL, adopting the global verbosity set by gl.set.verbosity(), or 2 if no global is set].

Details

The analysis is one-level only: the populations assigned in the pop slot are the single stratum. There is no provision for a nested hierarchy (for example regions containing populations). At least two populations are required.

A distance matrix supplied via the distance argument must cover exactly the individuals in x and must be labelled with the individual names (dimnames for a matrix, Labels for a dist object). The matrix is aligned to indNames(x) by those labels before use, so a matrix whose rows are ordered differently from x is reordered correctly; a matrix without labels, or with labels that do not match indNames(x), is a fatal error.

Significance is assessed by the one-tailed permutation test of pegas::amova(): the reported P.value is k/(permutations + 1), where k is the number of permuted variance components at least as large as the observed one, so a P.value of 0 is attainable. There is no seed argument; for a reproducible test, call set.seed() before calling this function. Set permutations = 0 to skip the test and return the variance components alone.

Value

An object of class 'amova' (package pegas): a list with a table (tab) of sums of square deviations (SSD), mean square deviations (MSD) and degrees of freedom; the variance components (varcomp; a data.frame of sigma2 and P.value when permutations > 0, otherwise a vector of sigma2); the variance coefficients (varcoef); and the matched call. Phi statistics are displayed when the object is printed.

Author(s)

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

Examples

```
#permutations should be higher, here set to 1 because of speed
if (isTRUE(getOption("dartR_fbm"))) bandicoot.gl <- gl.gen2fbm(bandicoot.gl)
out <- gl.amova(bandicoot.gl, permutations=1)
```

---

gl.assign.mahalanobis *Calculate probabilities of assignment of an individual of unknown provenance to populations based on Mahalanobis distance*

Description

Assigns an individual of unknown provenance to one or more target populations based on the unknown individual's distance from each population centroid in a PCA, measured as a squared Mahalanobis distance and tested against the spread of the population.

Usage

```
gl.assign.mahalanobis(
  x,
  unknown,
  nmin = 10,
  dim.limit = NULL,
  plevel = 0.001,
  n.best = NULL,
  verbose = NULL
)
```

Arguments

x	Name of the input genlight object [required].
unknown	Identity label of the focal individual whose provenance is unknown [required].
nmin	Minimum sample size for a target population to be included in the analysis [default 10].
dim.limit	Maximum number of dimensions to consider for the confidence envelope [default nPop(x) - 1].
plevel	Probability level below which a population is eliminated as a putative source [default 0.001].
n.best	If given a value, dictates the best n = n.best populations to retain for consideration (or more if there are ties). If not specified, the populations for which the probability is >= plevel are retained [default NULL].
verbose	Verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log; 3, progress and results summary; 5, full report [default 2, unless specified using gl.set.verbosity].

Details

The following process is followed:

1. Loci scored as missing in the unknown are removed, and remaining missing values are imputed (method "neighbour" of `gl.impute()`).
2. A PCA is undertaken on the populations to yield a series of orthogonal axes.
3. A workable subset of dimensions is chosen: dim.limit, or the number of leading eigenvalues larger than expected under the broken-stick model, whichever is the smaller (at least one).
4. The squared Mahalanobis distance (D2) of the unknown from the centroid of each population is calculated in those dimensions, using the population's own mean and covariance matrix.
5. D2 is tested with Hotelling's T-squared test for a new observation: with n individuals in the population and p dimensions, F = n (n - p) D2 / ((n + 1) p (n - 1)) follows an F distribution with p and n - p degrees of freedom. This test allows for the mean and covariance being estimated from a small sample; the chi-square approximation does not and rejects true source populations too often when n is small.

There are three considerations to assignment. First, consider only those populations for which the unknown has no excess of private alleles. This can be evaluated with `gl.assign.pa()`. Next, consider the PCA plot for the remaining populations and the position of the unknown in relation to their confidence ellipses, as produced by `gl.assign.pca()`. The third step (delivered by this function) considers the position of the unknown in relation to the confidence envelope in all selected dimensions of the ordination. The larger the probability, the greater the confidence in the assignment. If the unknown is an outlier with respect to a population, say at less than 0.001 probability, that population can be eliminated from further consideration.

Warning: gl.assign.mahalanobis() treats each selected dimension equally, without regard to the percentage variation explained. If the unknown is an outlier in a lower dimension with an explanatory variance of, say, 0.1%, the putative population will be eliminated. This is why the function only uses dimensions retained by the broken-stick criterion.

Populations with no more individuals than selected dimensions, or with a singular covariance matrix (for example when imputation makes individuals identical), cannot be tested. They are reported and retained.

Each of these approaches provides evidence, none are 100% definitive. They need to be interpreted cautiously.

Value

A genlight object containing the unknown individual (assigned to population "unknown") and the putative source populations, with their original (not imputed) genotypes. Loci scored as missing in the unknown and monomorphic loci are removed.

Author(s)

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

See Also

`gl.assign.pa`, `gl.assign.on.genotype`, `gl.assign.pca`

Other assignment: `gl.assign.on.genotype()`, `gl.assign.pa()`, `gl.assign.pca()`

Examples

```
# Focal individual from the Macleay River (EmmacMaclGeor)
test <- gl.assign.mahalanobis(testset.gl, unknown = "UC_00146", verbose = 3)
```

`gl.assign.on.genotype` *Use genotype to identify populations as possible source populations for an individual of unknown provenance*

Description

Identifies populations from which the individual of unknown provenance could have been drawn, given its genotype and the allele frequencies in the putative source populations. The putative source populations that survive are retained and returned in a genlight object.

Usage

```
gl.assign.on.genotype(
  x,
  unknown,
  nmin = 10,
  n.best = NULL,
  aic.threshold = 0.05,
  verbose = NULL
)
```

Arguments

`x`	Name of the input genlight object [required].
`unknown`	SpecimenID label (indName) 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].
`n.best`	If given a value, dictates the best n = n.best populations to retain for consideration (or more if there are ties) based on AIC weight. If not specified, then the putative source populations with AIC.wt >= aic.threshold are retained [default NULL].
`aic.threshold`	The critical value used to select populations for which there is some support as a putative source based on AIC weights [default 0.05].
`verbose`	Verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log; 3, progress and results summary; 5, full report [default 2, unless specified using gl.set.verbosity].

Details

For each putative source population, the function computes the log-likelihood of the unknown's multilocus genotype under Hardy-Weinberg equilibrium, treating loci as independent. Allele frequencies are estimated with the Rannala & Mountain (1997) prior, (x + 1/2) / (n + 1), where x is the count of the alternate allele and n the number of gene copies scored in the population. An allele absent from a population's sample therefore has a small non-zero frequency that depends on the sample size.

All populations are scored on the same loci: loci scored as missing in the unknown, or with no genotype in one or more of the retained populations, are excluded. Otherwise a population with more missing loci would sum fewer log-likelihood terms and appear more likely.

The log-likelihoods are converted to AIC values (-2 log-likelihood) and AIC weights. Because no parameters are fitted, the AIC weights equal the likelihoods normalised across populations, that is, the posterior probability of origin under equal prior probabilities. Populations with an AIC weight below aic.threshold are unlikely to be the source. The weights assume independent loci; with many linked loci they concentrate on the single best population more than the data justify.

A suitable estimate of the allele frequencies requires an adequate sample size, say >= 10.

A population named "unknowns" in the input is taken to hold other individuals of unknown provenance and is excluded from the reference populations.

WARNING: If a putative population is not in Hardy-Weinberg equilibrium, as might occur if it includes F1 hybrids and backcrosses, the genotype likelihoods are misspecified. For this reason, you may wish to remove populations that contain individuals likely to be subject to contemporary hybridization or admixture.

Value

A genlight object containing the focal individual (assigned to population "unknown") and the putative source populations based on AIC weights, with monomorphic loci removed. If no population qualifies, the genlight object contains only the unknown and a warning is given.

Author(s)

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

References

Rannala, B., & Mountain, J. L. (1997). Detecting immigration by using multilocus genotypes. Proceedings of the National Academy of Sciences, 94(17), 9197-9201.

See Also

gl.assign.pa, gl.assign.pca, gl.assign.mahalanobis

Other assignment: gl.assign.mahalanobis(), gl.assign.pa(), gl.assign.pca()

Examples

```
# Focal individual from the Macleay River (EmmacMaclGeor)
test <- gl.assign.on.genotype(testset.gl, unknown = "UC_00146",
                              nmin = 10, verbose = 3)
```

gl.assign.pa *Use private alleles to identify populations as possible source populations for an individual of unknown provenance*

Description

Identifies as putative source populations those for which the individual of unknown provenance has no more private alleles than expected. The putative source populations are retained and returned in a genlight object.

Usage

```
gl.assign.pa(
  x,
  unknown,
  nmin = 10,
  n.best = NULL,
  alpha = 0.01,
  verbose = NULL
)
```

Arguments

x	Name of the input genlight object [required].
unknown	SpecimenID label (indName) 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].
n.best	If given a value, dictates the best n = n.best populations to retain for consideration (or more if there are ties) based on private alleles. If not specified, then the putative source populations for which the count is within expectation (p >= alpha) are retained [default NULL].
alpha	The critical value used to select populations for which the unknown individual has a count of private alleles within expectation [default 0.01].
verbose	Verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log; 3, progress and results summary; 5, full report [default 2, unless specified using gl.set.verbosity].

Details

A private allele of the unknown with respect to a population is an allele (reference or alternate) carried by the unknown at a locus where no member of that population carries it. Loci with no genotype in the population, and loci scored as missing in the unknown, are not counted.

The expectation for each population is built from its own members: each member is compared with the remaining members of its population, and the counts of private alleles are log10(count + 1) transformed to approximate a normal distribution with a mean and standard deviation. The

count for the unknown is compared with this expectation, and populations for which the unknown has significantly more private alleles (one-tailed, p < alpha) are unlikely to be its source. If every member of a population has the same count (standard deviation zero or undefined), the population is retained when the unknown's count does not exceed that common value.

An excess of private alleles is evidence that the unknown does not belong to a population provided the sample size is adequate, say >= 10.

A population named "unknowns" in the input is taken to hold other individuals of unknown provenance and is excluded from the reference populations.

WARNING: If a putative population is not in Hardy-Weinberg equilibrium, as might occur if it includes F1 hybrids and backcrosses, then the standard deviation for the expectation will be inflated. This inflation may result in false identification of the population as a putative source for the focal unknown individual. For this reason, you may wish to remove populations that contain individuals likely to be subject to contemporary hybridization or admixture.

Value

A genlight object containing the focal individual (assigned to population "unknown") and the putative source populations, with loci scored as missing in the unknown and monomorphic loci removed. If no population qualifies, the genlight object contains only the unknown and a warning is given.

Author(s)

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

See Also

gl.assign.on.genotype, gl.assign.pca, gl.assign.mahalanobis

Other assignment: gl.assign.mahalanobis(), gl.assign.on.genotype(), gl.assign.pca()

Examples

```
# Focal individual from the Macleay River (EmmacMaclGeor)
test <- gl.assign.pa(testset.gl, unknown = "UC_00146", nmin = 10,
                     verbose = 3)
```

`gl.assign.pca` *Eliminate from consideration putative source populations for a specified individual of unknown provenance using PCA*

Description

Eliminates from consideration putative source populations for a specified individual of unknown provenance based on its position relative to the confidence ellipse of each putative source population in the top two dimensions of a PCA. Populations for which the unknown lies outside the specified confidence ellipse are eliminated.

Usage

```
gl.assign.pca(
  x,
  unknown,
  nmin = 10,
  plevel = 0.001,
  plot.out = TRUE,
  verbose = NULL
)
```

Arguments

x	Name of the input genlight object [required].
unknown	Identity label of the focal individual whose provenance is unknown [required].
nmin	Minimum sample size for a target population to be included in the analysis [default 10].
plevel	Alpha level for the bounding ellipses in the PCA plot [default 0.001].
plot.out	If TRUE, plot the 2D PCA showing the position of the unknown [default TRUE].
verbose	Verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log; 3, progress and results summary; 5, full report [default 2, unless specified using gl.set.verbosity].

Details

There are four approaches to population assignment of which this is one.

1. Eliminate those populations from consideration where the genotype of the unknown is not consistent with their allelic profiles. This can be evaluated with `gl.assign.on.genotype()`.
2. Eliminate those populations for which the unknown has substantial private alleles. Substantial numbers of private alleles are an indication that the unknown does not belong to a target population (provided that the sample size is adequate, say >= 10). This can be evaluated with `gl.assign.pa()`.
3. Consider the assignment probabilities using `gl.assign.mahalanobis()`. This approach calculates the squared Generalised Distance (Mahalanobis distance) of the unknown from the centroid of each remaining putative source population and tests it against the population's spread. This index takes into account the position of the unknown in relation to the confidence envelope in all selected dimensions of the ordination.
4. Consider the PCA plot for populations and the position of the unknown in relation to confidence ellipses as produced by `gl.assign.pca()`.

Each of these approaches is useful for decisions on which populations to eliminate as putative sources. They provide evidence for a decision on population assignment, but none are 100% definitive, and they need to be interpreted cautiously.

This function considers only the top two dimensions of the ordination. An unknown outside the ellipse in two dimensions has a two-dimensional squared Mahalanobis distance above the chi-square (2 df) critical value. Adding dimensions never reduces that distance, but the critical value also rises with the number of dimensions, so an unknown outside the ellipse in two dimensions may still

lie inside the envelope in more dimensions. Conversely, an unknown inside the ellipse in two dimensions may lie outside the envelope in deeper dimensions. Use a stringent plevel (e.g. the default of 0.001).

Loci scored as missing in the unknown are removed, and the remaining missing values are imputed (method "neighbour" of `gl.impute()`) before the PCA. Populations whose covariance matrix in the two dimensions is singular, or that have fewer than three individuals, cannot be tested; they are reported and retained.

Value

A genlight object containing the unknown individual (assigned to population "unknown") and the populations that are not eliminated, with their original (not imputed) genotypes. Loci scored as missing in the unknown are removed.

Author(s)

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

See Also

gl.assign.pa, gl.assign.on.genotype, gl.assign.mahalanobis

Other assignment: gl.assign.mahalanobis(), gl.assign.on.genotype(), gl.assign.pa()

Examples

```
if (requireNamespace("SIBER", quietly = TRUE) &&
    requireNamespace("directlabels", quietly = TRUE)) {
# Focal individual from the Macleay River (EmmacMaclGeor)
test <- gl.assign.pca(testset.gl, unknown = "UC_00146", verbose = 3)
}
```

---

gl.blast — *Aligns nucleotides sequences against those present in a target database using blastn*

Description

Basic Local Alignment Search Tool (BLAST; Altschul et al., 1990 & 1997) is a sequence comparison algorithm optimized for speed used to search sequence databases for optimal local alignments to a query. This function creates fasta files, creates databases to run BLAST, runs blastn and filters these results to obtain the best hit per sequence.

This function can be used to run BLAST alignment of short-read (DArTseq data) and long-read sequences (Illumina, PacBio... etc). You can use reference genomes from NCBI, genomes from your private collection, contigs, scaffolds or any other genetic sequence that you would like to use as reference.

Usage

```
gl.blast(
  x,
  ref_genome,
  task = "megablast",
  Percentage_identity = 70,
  Percentage_overlap = 0.8,
  bitscore = 50,
  number_of_threads = 2,
  verbose = NULL
)
```

Arguments

x	Either a genlight object containing a column named 'TrimmedSequence' containing the sequence of the SNPs (the sequence tag) trimmed of adapters as provided by DArT; or a path to a fasta file with the query sequences [required].
ref_genome	Path to a reference genome in fasta of fna format or in a compressed format ie "gz" extension [required].
task	Four different tasks are supported: 1) "megablast", for very similar sequences (e.g, sequencing errors), 2) "dc-megablast", typically used for inter-species comparisons, 3) "blastn", the traditional program used for inter-species comparisons, 4) "blastn-short", optimized for sequences less than 30 nucleotides [default 'megablast'].
Percentage_identity	Not a very sensitive or reliable measure of sequence similarity, however it is a reasonable proxy for evolutionary distance. The evolutionary distance associated with a 10 percent change in Percentage_identity is much greater at longer distances. Thus, a change from 80 to 70 percent identity might reflect divergence 200 million years earlier in time, but the change from 30 percent to 20 percent might correspond to a billion year divergence time change [default 70].
Percentage_overlap	Calculated as alignment length divided by the query length or subject length (whichever is shortest of the two lengths, i.e. length / min(qlen,slen)) [default 0.8].
bitscore	A rule-of-thumb for inferring homology, a bit score of 50 is almost always significant [default 50].
number_of_threads	Number of threads (CPUs) to use in blastn search [default 2].
verbose	Verbosity: 0, silent or fatal errors; 1, begin and end; 2, brief progress messages; 3, progress and results summary; 5, full report [default 2, unless specified using gl.set.verbosity].

Details

Installing BLAST

You can download the BLAST installs from: https://ftp.ncbi.nlm.nih.gov/blast/executables/blast+/LATEST/

The executables blastn and makeblastdb must be findable on the system PATH. BLAST cannot open a database whose path contains spaces, so the R temporary directory (tempdir()) must not contain spaces; if it does, set the environment variable TMPDIR (TMP on Windows) to a path without spaces and restart R.

Running BLAST

Four different tasks are supported:

- "megablast", for very similar sequences (e.g, sequencing errors)
- "dc-megablast", typically used for inter-species comparisons
- "blastn", the traditional program used for inter-species comparisons
- "blastn-short", optimized for sequences less than 30 nucleotides

If you are running a BLAST alignment of similar sequences, for example Turtle Genome Vs Turtle Sequences, the recommended parameters are: task = "megablast", Percentage_identity = 70, Percentage_overlap = 0.8 and bitscore = 50.

If you are running a BLAST alignment of highly dissimilar sequences because you are probably looking for sex linked hits in a distantly related species, and you are aligning for example sequences of Chicken Genome Vs Bassiana, the recommended parameters are: task = "dc-megablast", Percentage_identity = 50, Percentage_overlap = 0.01 and bitscore = 30.

Be aware that running BLAST might take a long time (i.e. days) depending of the size of your query, the size of your database and the number of threads selected for your computer.

BLAST output

The BLAST output is formatted as a table using output format 6, with columns defined in the following order:

- qseqid - Query Seq-id
- sacc - Subject accession
- stitle - Subject Title
- qseq - Aligned part of query sequence
- sseq - Aligned part of subject sequence
- nident - Number of identical matches
- mismatch - Number of mismatches
- pident - Percentage of identical matches
- length - Alignment length
- evalue - Expect value
- bitscore - Bit score
- qstart - Start of alignment in query
- qend - End of alignment in query
- sstart - Start of alignment in subject
- send - End of alignment in subject

- gapopen - Number of gap openings
- gaps - Total number of gaps
- qlen - Query sequence length
- slen - Subject sequence length
- PercentageOverlap - length / min(qlen,slen)

Three tables (all aligned sequences, the aligned sequences that pass the filters, and one hit per sequence) are saved as RDS files in tempdir(); their paths are printed when verbose >= 2. Each RDS holds a list of the call and the table.

BLAST filtering

BLAST output is filtered by ordering the hits of each sequence first by the highest percentage identity, then the highest percentage overlap and then the highest bitscore. Only one hit per sequence is kept based on these selection criteria.

Value

If the input is a genlight object: returns a genlight object with one hit per sequence merged to the slot $other$loc.metrics; sequences without a hit have NA in the BLAST columns. If the input is a fasta file: returns a dataframe with one hit per sequence. When no hit passes the filters, the genlight object is returned without BLAST columns, or an empty dataframe with the BLAST columns is returned for a fasta file.

Author(s)

Author(s): Berenice Talamantes Becerra & Luis Mijangos. Custodian: Luis Mijangos – Post to https://groups.google.com/d/forum/dartr

References

- Altschul, S. F., Gish, W., Miller, W., Myers, E. W., & Lipman, D. J. (1990). Basic local alignment search tool. Journal of molecular biology, 215(3), 403-410.
- Altschul, S. F., Madden, T. L., Schäffer, A. A., Zhang, J., Zhang, Z., Miller, W., & Lipman, D. J. (1997). Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. Nucleic acids research, 25(17), 3389-3402.
- Pearson, W. R. (2013). An introduction to sequence similarity ("homology") searching. Current protocols in bioinformatics, 42(1), 3-1.

See Also

gl.filter.locmetric

Examples

```
## Not run:
res <- gl.blast(x = testset.gl, ref_genome = "sequence.fasta")
# keep only loci with a confident hit
res <- gl.filter.locmetric(res, metric = "evalue", upper = 1e-5, lower = 0)
```

```
## End(Not run)
```

`gl.check.future.panel` *Evaluate SNP panel performance over future generations*

Description

Evaluates how well a SNP panel maintains its performance across multiple generations, either by simulating genetic drift internally (Wright-Fisher model) or by accepting user-supplied genlight objects representing the population at each future generation.

Usage

```
gl.check.future.panel(
  x,
  xorig,
  parameter = c("Fst", "He"),
  corr.method = c("spearman", "pearson"),
  ref = c("global", "by.pop"),
  metric = c("grm", "kinship"),
  inverse_dr = FALSE,
  sample = 1,
  Ne = NULL,
  neest.path = NULL,
  type = "drift",
  user.gl = NULL,
  n_gen = 10,
  replicates = 5,
  n_check = NULL,
  error.rate = 0.01,
  threshold = 0.001,
  n_sim_parents = 50,
  n_sim_hyb = 100,
  n_cores = 1,
  target = 0.9,
  plot.out = TRUE,
  plot.file = NULL,
  plot.dir = NULL,
  verbose = NULL
)
```

Arguments

x	A dartR or genlight object. The SNP panel (subset of loci from xorig).
xorig	A dartR or genlight object. The full original dataset. All loci in x must be present in xorig.

parameter	Character vector. Parameters to evaluate. Same options as gl.check.panel: "Fst", "He", "Ho", "Fis", "Nall", "Ne", "Ho_ind", "relatedness", "id", "parentage", "assignment", "hybridisation", "all", "all-Ne". For "hybridisation", all pairwise population crosses are evaluated when two or more populations are present. Default c("Fst", "He").
corr.method	Character. Correlation method passed to gl.check.panel for correlation-based performance metrics. Options are "spearman" (default) and "pearson".
ref	Character. Allele-frequency reference for "relatedness", passed to gl.check.panel. "global" (default) pools across the dataset using all pairs; "by.pop" uses within-population frequencies and within-population pairs only. Under type = "drift" the reference is recomputed from the drifted frequencies at each check generation.
metric	Character. Relatedness reporting scale passed to gl.check.panel: "grm" (default) or "kinship" (GRM/2). Does not change the correlation-based performance value.
inverse_dr	Logical. If TRUE, drift resistance performance is reported as 1 - drift_resistance (drift sensitivity), passed through to gl.check.panel. Default FALSE.
sample	Numeric in (0,1]. Proportion of individuals to randomly retain from each population of the panel at every check generation (the reference is always evaluated at full size). 1 (default) uses all individuals. Simulates genotyping fewer individuals with the panel. The proportion is maintained across populations.
Ne	Effective population size for the drift simulation (type = "drift" only). NULL (default) uses the census sizes from xorig (Ne = number of individuals per population). A single value sets a constant Ne for the largest population, with other populations scaled proportionally to their census fraction. A vector specifies an Ne trajectory (linearly interpolated to n_gen generations), enabling simulation of growing or declining populations; all populations grow or decline proportionally. The drift rate uses 2*Ne as the binomial size, decoupled from the number of individuals materialised at check generations.
neest.path	Character string. Path to NEstimator executable, required only when "Ne" is included in parameter.
type	Character. "drift" (default) simulates genetic drift internally. "user" accepts user-supplied genlights via user.gl.
user.gl	List of dartR or genlight objects. Required when type = "user". Element [[i]] is the full population dataset at generation i. Panel loci are extracted automatically.
n_gen	Integer. Number of generations to simulate (type = "drift" only). For type = "user", n_gen is set to length(user.gl). If n_check requests generations beyond n_gen, n_gen is extended automatically. Default 10.
replicates	Integer. Number of replicate drift simulations (type = "drift" only). Default 5.
n_check	Integer vector or NULL. Generations at which to evaluate panel performance. Generation 0 (baseline) is always included. NULL (default) = every generation. For type = "drift", n_gen is extended automatically to cover all values in n_check. For type = "user", values exceeding length(user.gl) are silently dropped.

error.rate	Numeric. Per-allele error rate. Default 0.01.
threshold	Numeric. Confusion threshold for "id" and "parentage". Default 0.001.
n_sim_parents	Integer. Simulated trios for "parentage". Default 50.
n_sim_hyb	Integer. Simulated F1 individuals per pairwise population cross for "hybridisation". Default 100.
n_cores	Integer or NULL. Cores for parallel parentage simulation, passed to gl.check.panel. Default 1 (sequential) to avoid nested parallelism when this function runs inside parallel workers. Set NULL for all-but-one core.
target	Numeric (0–1) or NULL. Reference line in the plot. Default 0.9.
plot.out	Logical. Print the performance-over-time plot. Default TRUE.
plot.file	Character string. File name for saving the plot.
plot.dir	Character string. Directory for saving the plot.
verbose	Integer. 0 = silent, 2 = full (default).

Details

Drift model: Allele frequencies tracked as K x L matrix. Each generation: $p' = \text{Binomial}(2N, p)/2N$ per population. No genlight objects created during drift — materialised only at check generations as proper "dartR" objects.

User mode: User supplies full-dataset genlights. Panel loci extracted from each element and gl.check.panel run at each check generation. For correlation-based parameters, the chosen corr.method is used at every generation.

Value

A list: $performance (data frame), $summary (per-generation summaries), $corr.method, $ref, $metric, $neest.path, $final_panels (panel genlights at last check generation), $plot.

Examples

```
# Short demonstration on four populations; use more replicates in real analyses
x <- gl.keep.pop(possums.gl, pop.list = levels(pop(possums.gl))[1:4], verbose = 0)
panel <- gl.select.panel(x, method = "random", nl = 20)

# check at generations 2 and 4; n_gen is extended automatically to 4
fut <- gl.check.future.panel(panel, x,
                             parameter = c("Fst", "He"),
                             n_gen  = 2,
                             replicates  = 2,
                             n_check = c(2, 4))
```

gl.check.panel *Check a SNP panel for specified conservation monitoring parameters*

Description

Evaluates how well a SNP panel recreates specified parameters of conservation concern compared to the full original dataset, or assesses the panel's power for individual identification, parentage assignment, population assignment, or F1 hybrid detection. Multiple parameters can be evaluated simultaneously. All plots are combined on a single page with the summary bar chart as the first panel in the same grid.

Usage

```
gl.check.panel(
  x,
  xorig = NULL,
  parameter = "Fst",
  neest.path = NULL,
  plot.out = TRUE,
  plot.file = NULL,
  plot.dir = NULL,
  verbose = NULL,
  corr.method = c("spearman", "pearson"),
  ref = c("global", "by.pop"),
  metric = c("grm", "kinship"),
  inverse_dr = FALSE,
  sample = 1,
  error.rate = 0.01,
  threshold = 0.001,
  n_sim_parents = 100,
  n_sim_hyb = 100,
  n_cores = 1,
  target = 0.9
)
```

Arguments

x	A dartR or genlight object containing the SNP panel (subset of loci).
xorig	A dartR or genlight object containing the full original dataset for comparison, with the same individuals (by name) and populations as x. Default NULL. Required for all correlation/comparison parameters ("Fst", "He", "Ho", "Fis", "Nall", "Ne", "Ho_ind", "Fis_ind", "relatedness"). May be left NULL when only "drift_resistance", "id", "parentage", "assignment", or "hybridisation" are requested, as these are evaluated on x alone.
parameter	Character string or vector specifying the parameter(s) to evaluate. Options: • "Fst" — pairwise FST' (between-population).

- "He" — expected heterozygosity per population.
- "Ho" — observed heterozygosity per population.
- "Fis" — inbreeding coefficient per population.
- "Nall", "Na" or "Ar" — rarefaction-corrected allelic richness per population.
- "Ne" — effective population size per population.
- "Ho_ind" — individual observed heterozygosity. Correlation r² of per-individual Ho between the panel and full dataset, ignoring population membership.
- "Fis_ind" — individual within-population inbreeding coefficient. For each individual, $F_i = 1 - H_{O,i}/H_{E,i}$ is calculated from the full dataset and from the panel. Performance is the selected corr.method squared correlation between the two sets of individual estimates, consistent with the other correlation-based parameters.
- "drift_resistance" — absolute panel score from 0 to 1 based on within-population minor allele frequencies. For locus l in population k, the score is $[2\min(p_{lk}, 1-p_{lk})]^2$. Scores are averaged across loci within each population and then equally across populations. A score of 1 means every panel locus has allele frequency 0.5 in every population; 0 means all panel loci are fixed.
- "relatedness" — pairwise genomic relatedness (VanRaden GRM), optionally rescaled to kinship via metric. The allele-frequency reference is controlled by ref: pooled across the dataset ("global", population membership ignored) or computed within each population ("by.pop", within-population pairs only).
- "id" — individual identification power.
- "parentage" — parentage assignment power.
- "assignment" — population assignment accuracy (LOO).
- "hybridisation" — F1 hybrid detection across all pairwise population crosses (requires at least 2 populations in x).
- "all" — all parameters except "hybridisation".
- "all-Ne" — all except "Ne" and "hybridisation".

neest.path — Character string. Path to NEstimator executable, required only for "Ne".

plot.out — Logical. Print combined plot. Default TRUE.

plot.file — Character string. File name for saving the plot.

plot.dir — Character string. Directory for saving the plot.

verbose — Integer. 0 = silent, 1 = parameter names and performance scores only, 2 = full progress (default).

corr.method — Character. Correlation method used as the performance indicator for correlation-based parameters. Options are "spearman" (default, preserving previous behaviour) and "pearson". Applies to "Fst", "He", "Ho", "Fis", "Nall", "Ne", "Ho_ind", "Fis_ind", and "relatedness". Accuracy/proportion metrics such as "id", "parentage", "assignment", and "hybridisation" are unchanged.

ref	Character. Allele-frequency reference for "relatedness". "global" (default) pools allele frequencies across the whole dataset and uses all pairs; between differentiated populations this produces strongly negative values that mainly reflect structure. "by.pop" computes allele frequencies within each population, references each population to its own base, and retains only within-population pairs — appropriate for within-population kinship. Populations with fewer than two individuals are skipped with a warning. Ignored by all other parameters.
metric	Character. Reporting scale for "relatedness". "grm" (default) returns the VanRaden genomic relationship matrix; "kinship" returns kinship = GRM / 2 (so an outbred self is ~0.5, parent–offspring/full-sib ~0.25). Because this is a constant rescaling applied to both panel and full dataset, it does not change the correlation-based performance value; it affects only the reported values, axis labels, and plot. Ignored by all other parameters.
inverse_dr	Logical. If TRUE and "drift_resistance" is among the requested parameters, the performance is reported as 1 - drift_resistance (drift sensitivity). A high value then indicates a panel whose loci are near fixation and therefore maximally responsive to allele-frequency change — desirable for genetic-erosion monitoring but not for long-lived stable panels. Default FALSE.
sample	Numeric in (0,1]. Proportion of individuals to randomly retain from each population of the *panel* before evaluation. 1 (default) uses all individuals; 0.5 keeps a random half of each population. The reference dataset xorig always uses all individuals, so this tests how panel performance degrades when fewer individuals are genotyped with the panel while the truth is known from the full dataset. The proportion is maintained across all populations.
error.rate	Numeric. Per-allele genotyping error rate. Default 0.01.
threshold	Numeric. Confusion threshold for "id" and "parentage". Default 0.001.
n_sim_parents	Integer. Simulated trios for "parentage". Default 100.
n_sim_hyb	Integer. Simulated F1 individuals per pairwise population cross for "hybridisation". Total F1 simulations are n_sim_hyb * choose(nPop(x), 2). Default 100.
n_cores	Integer or NULL. Cores for parallel parentage simulation. Default 1 (sequential), which prevents accidental nested parallelism when this function is called from inside parallel workers (e.g. gl.select.panel.combined, parallel chains, or study scripts). Set to NULL to use all-but-one core, or an explicit integer. On Windows always sequential.
target	Numeric (0–1) or NULL. Reference line in the summary bar chart. Default 0.9.

Details

Performance metric (0–1):

- Population and between-population parameters ("Fst", "He", "Ho", "Fis", "Nall", "Ne") — chosen corr.method r^2 between panel and full dataset values. Plots show both Spearman r^2 and Pearson R^2.
- "Ho_ind" — chosen corr.method r^2 of per-individual observed heterozygosity. Population membership is ignored.

- "Fis_ind" — agreement between individual inbreeding estimates from the panel and the full dataset. Within each population, expected heterozygosity is calculated as $H_{e,l} = 2p_l(1 - p_l)$. For each individual and its called loci, $F_i = 1 - H_{O,i}/H_{E,i}$. The returned data contain the actual fis_orig and fis_panel values for every individual. Performance is the chosen corr.method squared correlation between fis_orig and fis_panel, exactly as for the other correlation-based parameters. The summary also reports RMSE, MAE, bias, Pearson R^2, and Spearman r^2.
- "drift_resistance" — absolute panel score $\mathrm{mean}_k[\mathrm{mean}_l\{(2\min(p_{lk}, 1 - p_{lk}))^2\}]$. This is not a correlation with xorig; it directly rewards loci with intermediate allele frequencies and strongly penalises rare or nearly fixed alleles.
- "relatedness" — chosen corr.method r² of all retained pairwise relatedness values (upper triangle), on the scale set by metric (GRM or kinship = GRM/2) and the reference set by ref ("global": all pairs, pooled frequencies; "by.pop": within-population pairs, local frequencies) (VanRaden 2008).
- "id" — fraction of individuals with MaxPmatch below threshold.
- "parentage" — fraction of simulated trios correctly assigned.
- "assignment" — fraction correctly assigned by LOO.
- "hybridisation" — correct F1-pair assignment rate, calculated as the proportion of simulated F1 hybrids assigned to the correct pairwise F1 class across all pairwise population crosses.

Heterozygosity caching: When any of "He", "Ho", "Fis" are requested, gl.report.heterozygosity is called once per dataset and reused for all three.

Relatedness reference and scale: The VanRaden GRM is centred on a reference allele-frequency vector, so a value of zero corresponds to the average pair in that reference and roughly half of the off-diagonal values are negative by construction. With ref = "global" the reference is pooled across all populations; in structured data this makes between-population pairs strongly negative, so the metric largely tracks recovery of population structure. With ref = "by.pop" each population is referenced to its own allele frequencies and only within-population pairs are kept, giving within-population kinship; this requires adequate within-population sample sizes. metric = "kinship" rescales the matrix by 1/2 (numerator-relationship to kinship); as a constant applied to both datasets it leaves the correlation performance unchanged. The panel relatedness data frame columns are rel_orig and rel_panel.

Value

A named list with one element per parameter, each containing: name, performance, corr.method, data, summary, confusion, and plot. corr.method is the correlation method used for correlation-based parameters and NA otherwise. For most parameters, summary and confusion are NULL; for "drift_resistance", summary contains scores by population; for "Fis_ind", summary contains correlation and error diagnostics; for "hybridisation", summary gives per-cross F1 detection and correct-pair rates, and confusion gives the full true-cross by assigned-class table. The performance value for "hybridisation" is the correct F1-pair rate.

References

VanRaden, P.M. (2008). Efficient methods to compute genomic predictions. *Journal of Dairy Science*, 91(11), 4414–4423.

Examples

```
# Short demonstration on four populations
x <- gl.keep.pop(possums.gl, pop.list = levels(pop(possums.gl))[1:4], verbose = 0)
panel <- gl.select.panel(x, method = "random", nl = 20)
res   <- gl.check.panel(panel, x, parameter = "all-Ne")
sapply(res, function(r) r$performance)
```

`gl.collapse` *Collapses a distance matrix by amalgamating populations with pairwise fixed difference count less than or equal to a threshold*

Description

This function takes the list of matrices generated by gl.fixed.diff() and amalgamates populations with a fixed difference count less than or equal to a specified threshold (`tpop`).

The function then applies the new population assignments to the genlight object and recalculates the fixed difference and associated matrices.

Usage

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

Arguments

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

Details

Populations are grouped transitively: A and C are amalgamated when A-B and B-C each have `tpop` or fewer fixed differences, even if A-C has more. Each group is labelled with the first population name in alphabetical order followed by "+". Population pairs with no fixed difference count (NA, e.g. no loci in common) are not amalgamated.

`tloc` must be the value used in the call to gl.fixed.diff() that produced `fd`; the function checks this by recomputing the fixed differences and stops if they differ.

If no populations are amalgamated, `fd` is returned unchanged. If all populations fall into one group, the returned genlight has one population and the matrices are 1 x 1.

Value

A list of class 'fd' containing the new genlight object and square matrices:

1. $gl – the new genlight object with populations collapsed;
2. $fd – raw fixed differences;
3. $pcfd – percent fixed differences;
4. $nobs – mean no. of individuals used in each comparison;
5. $nloc – total number of loci used in each comparison;
6. $expfpos – NA's, populated by gl.fixed.diff [by simulation];
7. $sdfpos – NA's, populated by gl.fixed.diff [by simulation];
8. $pval – NA's, populated by gl.fixed.diff [by simulation].

Author(s)

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

Examples

```
fd <- gl.fixed.diff(testset.gl, tloc = 0.05)
fd
fd2 <- gl.collapse(fd, tpop = 1, tloc = 0.05)
fd2
fd3 <- gl.collapse(fd2, tpop = 1, tloc = 0.05)
fd3

 fd <- gl.fixed.diff(testset.gl, tloc = 0)
 fd2 <- gl.collapse(fd)
```

---

gl.evanno *Creates an Evanno plot from a STRUCTURE run object*

Description

This function takes a structure run object (output from gl.run.structure), computes the Evanno et al. (2005) statistics used to choose the number of clusters K, and plots them.

Usage

```
gl.evanno(
  sr,
  plot.out = TRUE,
  plot.theme = theme_dartR(),
  plot.dir = NULL,
  plot.file = NULL,
  verbose = NULL
)
```

Arguments

sr	Structure run object from gl.run.structure [required].
plot.out	TRUE: the plots are shown. FALSE: the plots are returned as ggplot objects but not shown [default TRUE].
plot.theme	Theme for the plots. See Details for options [default theme_dartR()].
plot.dir	Directory in which to save files [default = tempdir(), unless specified using gl.set.wd].
plot.file	Name for the RDS binary file to save the combined plot (base name only, exclude extension) [default NULL].
verbose	Verbosity: 0, silent or fatal errors; 1, begin and end; 2, brief progress messages; 3, progress and results summary; 5, full report [default 2, unless specified using gl.set.verbosity].

Details

For each K the function takes the mean and standard deviation of the estimated Ln probability of the data, LnP(K), over the replicate runs, and computes LnP'(K) = LnP(K) - LnP(K - 1), |LnP''(K)| = |LnP'(K + 1) - LnP'(K)| and delta K = |LnP''(K)| / sd(LnP(K)). The differences are taken on the mean LnP(K) of each K, as in the strataG package (Archer et al. 2016) from which the code was adapted. The K with the largest delta K is the usual choice.

The method needs at least three consecutive values of K and at least two replicates per K. Delta K is NA for the smallest and largest K, for K values whose neighbours K - 1 or K + 1 were not run, for K with a single replicate, and for K whose replicates all report the same LnP(K) (sd = 0); a warning names these K at verbose >= 1.

Examples of other themes that can be used can be consulted in

- https://ggplot2.tidyverse.org/reference/ggtheme.html and
- https://yutannihilation.github.io/allYourFigureAreBelongToUs/ggthemes/

Value

A list with two elements: df, a data frame with one row per K and columns k, reps (number of replicates), mean.ln.k, sd.ln.k, ln.pk (LnP'(K)), ln.ppk (|LnP''(K)|) and delta.k; and plots, a list of ggplot objects mean.ln.k, ln.pk, ln.ppk, delta.k (only when delta K could be computed for at least one K) and combined (all panels in one figure).

Author(s)

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

References

- Pritchard, J.K., Stephens, M., Donnelly, P. (2000) Inference of population structure using multilocus genotype data. Genetics 155, 945-959.

- Archer, F. I., Adams, P. E. and Schneiders, B. B. (2016) strataG: An R package for manipulating, summarizing and analysing population genetic data. Mol Ecol Resour. doi:10.1111/1755-0998.12559
- Evanno, G., Regnaut, S., and J. Goudet. 2005. Detecting the number of clusters of individuals using the software STRUCTURE: a simulation study. Molecular Ecology 14:2611-2620.

See Also

gl.run.structure, gl.plot.structure

Other population structure: gl.map.popcluster(), gl.map.snmf(), gl.map.structure(), gl.plot.faststructure(), gl.plot.popcluster(), gl.plot.snmf(), gl.plot.structure(), gl.read.structure(), gl.run.faststructure(), gl.run.popcluster(), gl.run.snmf()

Examples

```
# examples need structure to be installed on the system (see above)
## Not run:
 bc <- bandicoot.gl[,1:100]
 sr <- gl.run.structure(bc, k.range = 2:5, num.k.rep = 3, exec = './structure.exe')
 ev <- gl.evanno(sr)
 ev$df
 qmat <- gl.plot.structure(sr, K=3)
 head(qmat)
 gl.map.structure(qmat, bc, K=3, scalex=1, scaley=0.5)

## End(Not run)
```

---

gl.find.genes.for.loci

Map loci (SNPs) to the nearest gene feature from a GFF

Description

Given a SNP genlight object and a GFF3 annotation file, find the closest gene (or transcript, if requested) for each input locus. If a locus falls within a gene interval, that gene is considered the closest with distance 0.

Usage

```
gl.find.genes.for.loci(
  x,
  gff.file,
  loci = NULL,
  include_types = c("gene", "pseudogene"),
  fallback_to_mrna = TRUE,
  save2tmp = FALSE,
  verbose = NULL
)
```

Arguments

x	A SNP genlight object with mapped loci. Must contain per-locus x$chromosome and x$position. [required]
gff.file	Path to a GFF3 file, plain or gzip-compressed. A path without the .gz extension is also accepted when only the .gz file exists. [required]
loci	Character vector of locus names to map, matching locNames(x). [default NULL, all loci with a chromosome and position]
include_types	Character vector of GFF types to treat as "gene" features. [default c("gene","pseudogene")]
fallback_to_mrna	Logical. If no rows match include_types, use transcript features c("mRNA","transcript") as proxies. [default TRUE]
save2tmp	Logical: save the result table to tempdir() (retrievable with gl.list.reports / gl.print.reports). [default FALSE]
verbose	Verbosity: 0, silent or fatal errors; 1, begin and end; 2, brief progress messages; 3, progress and results summary; 5, full report [default 2, unless specified using gl.set.verbosity].

Details

The function parses common keys in the GFF attributes column (ID, Name, gene, product, Parent, gene_symbol) to provide informative gene labels. When a gene feature carries no product attribute, the product of its child features (mRNA, then CDS or exon) is used, which is where NCBI and Ensembl GFF3 files place it. A trailing ", transcript variant ..." label is removed from such a product, because it names one transcript rather than the gene.

Closeness is measured on the same sequence (chromosome/contig) as:

- 0 if the locus is within [gene_start, gene_end]
- otherwise, the minimum bp distance to the interval edges

If multiple genes are exactly equally close, a deterministic tie-break is applied: closest to gene midpoint, then shorter gene length, then lexicographic gene_id. A locus on a sequence that has no gene feature is returned with NA in the gene columns.

The sequence names in x$chromosome must match the first column of the GFF. A name of the form <accession>_<description>, as DArT reports reference sequences (e.g. NC_041728.1_chromosome_1), is matched to the GFF sequence it starts with (NC_041728.1), and the chrom column then holds the GFF name. Loci without a chromosome name or a position, and sequence names absent from the GFF, are reported at verbose >= 1. GFF3 percent-encoding in gene names, symbols and products is decoded (e.g. %2C to a comma).

Value

A data.table with one row per locus that has a chromosome and position, and columns: locus, chrom, pos, gene_start, gene_end, gene_type, gene_id, gene_name, gene_symbol, gene_product, gene_attributes, distance_bp, nearest_side, gene_strand, relative_position. ‘distance_bp‘ is the absolute distance in bp; ‘nearest_side‘ is "inside", "left" (locus < gene_start) or "right" (locus > gene_end) in coordinate space; ‘relative_position‘ is "inside", "upstream" or "downstream" with respect to the gene strand (NA when the strand is unknown).

Author(s)

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

See Also

Other annotation and mapping helpers: gl.find.loci.in.genes()

Examples

```
# A small GFF3 with two genes on one sequence
gff <- tempfile(fileext = ".gff")
writeLines(c(
  "##gff-version 3",
  paste("chr1", "src", "gene", 100, 200, ".", "+", ".",
        "ID=gene-A;Name=A", sep = "\t"),
  paste("chr1", "src", "gene", 500, 800, ".", "-", ".",
        "ID=gene-B;Name=B", sep = "\t")
), gff)
x <- testset.gl
x@chromosome <- factor(c("chr1", "chr1", rep(NA, nLoc(x) - 2)))
x@position <- c(150L, 300L, rep(NA_integer_, nLoc(x) - 2))
gl.find.genes.for.loci(x, gff.file = gff, loci = locNames(x)[1:2])
```

---

gl.find.loci.in.genes *Find loci that fall within genes matching a pattern (from a GFF)*

Description

Given a genlight object with mapped loci (chromosome and position) and a gene annotation file (GFF3, plain or gzip-compressed), this function returns the names of the loci that fall inside genes whose annotation matches a pattern (e.g. "MHC", "major histocompatibility").

Usage

```
gl.find.loci.in.genes(x, gff.file, gene, save2tmp = FALSE, verbose = NULL)
```

Arguments

x	Name of the genlight object containing SNP or SilicoDArT data, with per-locus x$chromosome and x$position [required].
gff.file	Path to a GFF3 file, plain or gzip-compressed. A path without the .gz extension is also accepted when only the .gz file exists [required].
gene	Regular expression identifying the target genes, e.g. "(?i)major histocompatibility|\bMHC\b" [required].
save2tmp	Logical: save the table of loci and the genes they fall in to tempdir() (retrievable with gl.list.reports and gl.print.reports) [default FALSE].

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

Details

The pattern is tested against the whole attributes column of every GFF row. A gene or pseudogene is selected when its own row matches, or when any of its descendant features matches (mRNA, lnc_RNA, CDS, exon and so on, followed up through their Parent attribute, or linked by the gene= key). NCBI and Ensembl GFF3 files put the product description on these child rows, not on the gene row.

Because the whole attribute string is searched, a short pattern can match more than intended: "TAP" also matches WTAP, METAP1 or STAP1, and "ID" matches every feature. Use word boundaries (e.g. "\bTAP1\b") and "(?i)" for case-insensitive matching.

Loci are placed by x$chromosome and x$position, which must be the sequence name used in the first column of the GFF and a position in genome coordinates. dartR sets x$position to the SNP position within the tag when the data are read, so replace it with the genome position first (e.g. the ChromPos locus metric plus SnpPosition). A sequence name of the form <accession>_<description>, as DArT reports reference sequences (e.g. NC_041728.1_chromosome_1), is matched to the GFF sequence it starts with (NC_041728.1). Loci without a chromosome name or a position, and sequence names absent from the GFF, are reported at verbose >= 1.

To learn which gene each locus falls in, pass the result to gl.find.genes.for.loci.

Value

A character vector of the names of the loci that fall inside the matching genes, in genome order.

Author(s)

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

See Also

Other annotation and mapping helpers: gl.find.genes.for.loci()

Examples

```
# A small GFF3 with one MHC gene and one other gene on one sequence
gff <- tempfile(fileext = ".gff")
writeLines(c(
  "##gff-version 3",
  paste("chr1", "src", "gene", 100, 200, ".", "+", ".",
        "ID=gene-A;Name=A", sep = "\t"),
  paste("chr1", "src", "mRNA", 100, 200, ".", "+", ".",
        "ID=rna-A;Parent=gene-A;product=MHC class I antigen", sep = "\t"),
  paste("chr1", "src", "gene", 500, 800, ".", "-", ".",
        "ID=gene-B;Name=B;description=kinase", sep = "\t")
), gff)
x <- testset.gl
```

```
x@chromosome <- factor(c("chr1", "chr1", rep(NA, nLoc(x) - 2)))
x@position <- c(150L, 600L, rep(NA_integer_, nLoc(x) - 2))
gl.find.loci.in.genes(x, gff.file = gff,
                      gene = "(?i)major histocompatibility|\\bMHC\\b")
```

`gl.ld.distance` *Plots linkage disequilibrium against distance by population*

Description

The function bins SNP pairs by the distance between them, in base pairs pooled over all chromosomes, and plots the mean pairwise LD of each bin for each population, with a red line at R.squared = 0.2, the threshold commonly used to imply that two loci are unlinked (Delourme et al., 2013; Li et al., 2014).

Usage

```
gl.ld.distance(
  ld.report,
  ld.resolution = 1e+05,
  pop.colors = NULL,
  plot.title = " ",
  plot.theme = NULL,
  plot.out = TRUE,
  plot.file = NULL,
  plot.dir = NULL,
  verbose = NULL
)
```

Arguments

`ld.report`	Output from function `gl.report.ld.map` [required].
`ld.resolution`	Width of the distance bins in number of base pairs [default 100000].
`pop.colors`	A palette function (e.g. `rainbow`) or a vector with at least as many colors as there are populations in the dataset [default NULL].
`plot.title`	Title of the plot [default " "].
`plot.theme`	User specified theme [default NULL].
`plot.out`	Specify if plot is to be produced [default TRUE].
`plot.file`	Name for the RDS binary file to save (base name only, exclude extension) [default NULL]
`plot.dir`	Directory in which to save files [default = tempdir(), unless specified using gl.set.wd].
`verbose`	Verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log; 3, progress and results summary; 5, full report [default 2, unless specified using gl.set.verbosity].

Details

Bins are (1, 1 + ld.resolution], (1 + ld.resolution, 1 + 2 * ld.resolution], ..., with the first bin closed on the left and the last bin ending at the largest distance in `ld.report`. Each point is the mean LD of the pairs in a bin and is plotted at the upper edge of the bin.

The red threshold line applies to R.squared, the default LD measure of `gl.report.ld.map`. If the report was computed with another measure (e.g. D.prime), the line has no meaning.

Value

A data.table with one row per population and bin: pop, distance (upper edge of the bin in base pairs), ld.stat (mean LD of the pairs in the bin; NA for an empty bin) and n.pairs (number of pairs with an LD value in the bin).

Author(s)

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

References

- Delourme, R., Falentin, C., Fomeju, B. F., Boillot, M., Lassalle, G., Andre, I., . . . Marty, A. (2013). High-density SNP-based genetic map development and linkage disequilibrium assessment in Brassica napusL. BMC genomics, 14(1), 120.
- Li, X., Han, Y., Wei, Y., Acharya, A., Farmer, A. D., Ho, J., . . . Brummer, E. C. (2014). Development of an alfalfa SNP array and its use to evaluate patterns of population structure and linkage disequilibrium. PLoS One, 9(1), e84329.

See Also

Other ld functions: `gl.ld.haplotype()`

Examples

```
# gl.report.ld.map (dartR.base) needs snpStats and fields
if (requireNamespace("snpStats", quietly = TRUE) &&
    requireNamespace("fields", quietly = TRUE)) {
  require("dartR.data")
  x <- platypus.gl
  x <- gl.filter.callrate(x, threshold = 1)
  x <- gl.filter.monomorphs(x)
  x$position <- x$other$loc.metrics$ChromPos_Platypus_Chrom_NCBIv1
  x$chromosome <- as.factor(x$other$loc.metrics$Chrom_Platypus_Chrom_NCBIv1)
  ld_res <- gl.report.ld.map(x, ld.max.pairwise = 10000000)
  ld_res_2 <- gl.ld.distance(ld_res, ld.resolution = 1000000)
}
```

`gl.ld.haplotype` *Visualize patterns of linkage disequilibrium and identification of haplotypes*

Description

This function plots a Linkage disequilibrium (LD) heatmap, where the colour shading indicates the strength of LD. Chromosome positions (Mbp) are shown on the horizontal axis, and haplotypes appear as triangles and delimited by dark yellow vertical lines. Numbers identifying each haplotype are shown in the upper part of the plot.

The heatmap also shows heterozygosity for each SNP.

When `haplo_id = TRUE`, the function identifies haplotypes as runs of adjacent SNPs whose pairwise LD is at least `ld_threshold_haplo` and that contain at least `min_snps` SNPs. With the default `haplo_id = FALSE` no haplotypes are identified and the returned table is empty.

Usage

```
gl.ld.haplotype(
  x,
  pop_name = NULL,
  chrom_name = NULL,
  ld_max_pairwise = 1e+07,
  maf = 0.05,
  ld_stat = "R.squared",
  ind.limit = 10,
  haplo_id = FALSE,
  min_snps = 10,
  ld_threshold_haplo = 0.5,
  plot_het = TRUE,
  snp_pos = TRUE,
  target.snp1 = NULL,
  target.snp2 = NULL,
  target.snp3 = NULL,
  col.all = "black",
  col.target1 = "green",
  col.target2 = "blue",
  col.target3 = "red",
  coordinates = NULL,
  color_haplo = "viridis",
  color_het = "deeppink",
  plot.out = TRUE,
  plot.save = FALSE,
  plot.dir = NULL,
  verbose = NULL
)
```

Arguments

x	Name of the genlight object containing the SNP data [required].
pop_name	Name of the population to analyse. If NULL all the populations are analysed [default NULL].
chrom_name	Name of the chromosome to analyse. If NULL all the chromosomes are analysed [default NULL].
ld_max_pairwise	Maximum distance in number of base pairs at which LD should be calculated [default 10000000].
maf	Minor allele frequency (by population) threshold to filter out loci. If a value > 1 is provided it will be interpreted as MAC (i.e. the minimum number of times an allele needs to be observed) [default 0.05].
ld_stat	The LD measure to be calculated: "LLR", "OR", "Q", "Covar", "D.prime", "R.squared", and "R". See ld (package snpStats) for details [default "R.squared"].
ind.limit	Minimum number of individuals that a population must have to be analysed. Populations with fewer individuals are skipped, with a warning; the function stops if every population is skipped [default 10].
haplo_id	Whether to identify haplotypes [default FALSE].
min_snps	Minimum number of SNPs that a haplotype must contain to be called [default 10].
ld_threshold_haplo	Minimum LD between adjacent SNPs to call a haplotype [default 0.5].
plot_het	Whether to plot heterozygosity [default TRUE].
snp_pos	Whether to plot SNP positions. The SNP position track is drawn only when no haplotypes are identified [default TRUE].
target.snp1	Vector of position(s) of target SNP(s) in base pairs; the closest SNP to each position is highlighted in the SNP position track [default NULL].
target.snp2	Vector of position(s) of target SNP(s) in base pairs [default NULL].
target.snp3	Vector of position(s) of target SNP(s) in base pairs [default NULL].
col.all	Color of line indicating position for all SNPs [default "black"].
col.target1	Color of line indicating position for target.snp1 [default "green"].
col.target2	Color of line indicating position for target.snp2 [default "blue"].
col.target3	Color of line indicating position for target.snp3 [default "red"].
coordinates	A vector of two elements with the start and end coordinates in base pairs to which restrict the analysis e.g. c(1,1000000) [default NULL].
color_haplo	Color palette for haplotype plot. Options are: "magma", "inferno", "plasma", "viridis", "cividis", "rocket", "mako" and "turbo" [default "viridis"].
color_het	Color for heterozygosity [default "deeppink"].
plot.out	Specify if heatmap plot is to be produced [default TRUE].
plot.save	Whether to save each plot in pdf format, as <population>_<chromosome>.pdf in plot.dir [default FALSE].

plot.dir	Directory in which to save files [default tempdir(), or the directory set with gl.set.wd()].
verbose	Verbosity: 0, silent or fatal errors; 1, begin and end; 2, brief progress messages; 3, progress and results summary; 5, full report [default 2, unless specified using gl.set.verbosity].

Details

The information for SNP's position should be stored in the genlight accessor "@position" and the SNP's chromosome name in the accessor "@chromosome" (see examples). The function will then calculate LD within each chromosome. Chromosomes with fewer than four SNPs after filtering are skipped.

The output of the function includes a table with the haplotypes that were identified and their location.

Colors of the heatmap (color_haplo) are based on the function scale_fill_viridis from package viridis. Other color palettes options are "magma", "inferno", "plasma", "viridis", "cividis", "rocket", "mako" and "turbo".

Value

A data frame with one row per haplotype identified: population, chromosome, haplotype number, start and end (base pairs), the corresponding x coordinates in the LD plot, midpoints and a label in Mbp. Returned invisibly; empty when haplo_id = FALSE or no haplotype is found.

Author(s)

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

See Also

Other ld functions: gl.ld.distance()

Examples

```
require("dartR.data")
x <- platypus.gl
x <- gl.filter.callrate(x, threshold = 1)
# only the first 15 individuals because of speed during tests
x <- gl.keep.pop(x, pop.list = "TENTERFIELD")[1:15, ]
x$chromosome <- as.factor(x$other$loc.metrics$Chrom_Platypus_Chrom_NCBIv1)
x$position <- x$other$loc.metrics$ChromPos_Platypus_Chrom_NCBIv1
ld_res <- gl.ld.haplotype(x,
  chrom_name = "NC_041728.1_chromosome_1",
  ld_max_pairwise = 10000000
)
```

gl.LDNe *Estimates effective population size using the Linkage Disequilibrium method based on NeEstimator (V2)*

Description

This function is basically a convenience function that runs the LD Ne estimator using Neestimator2 http://www.molecularfisherieslaboratory.com.au/neestimator-software/ within R using the provided genlight object. To be able to do so, the software has to be downloaded from their website and the appropriate executable Ne2-1 has to be copied into the path as specified in the function (see example below).

Usage

```
gl.LDNe(
  x,
  outfile = "genepopLD.txt",
  outpath = tempdir(),
  neest.path = getwd(),
  critical = 0,
  singleton.rm = TRUE,
  mating = "random",
  pairing = "all",
  Waples.correction = NULL,
  Waples.correction.value = NULL,
  naive = FALSE,
  plot.out = TRUE,
  plot_theme = theme_dartR(),
  plot_colors_pop = gl.select.colors(x, verbose = 0),
  plot.file = NULL,
  plot.dir = NULL,
  verbose = NULL
)
```

Arguments

x	Name of the genlight object containing the SNP data [required].
outfile	File name of the output file with all results from Neestimator 2 [default 'genepopLD.txt'].
outpath	Path where to save the output file. Use outpath=getwd() or outpath='.' when calling this function to direct output files to your working directory [default tempdir(), mandated by CRAN].
neest.path	Path to the folder of the NE2-1 file. Please note there are 3 different executables depending on your OS: Ne2-1.exe (=Windows), Ne2-1M (=Mac), Ne2-1L (=Linux). You only need to point to the folder (the function will recognise which OS you are running) [default getwd()].

`critical` (vector of) Critical values that are used to remove alleles based on their minor allele frequency. This can be done before using the gl.filter.maf function, therefore the default is set to 0 (no loci are removed). To run for MAF 0 and MAF 0.05 at the same time specify: critical = c(0,0.05) [default 0].

`singleton.rm` Whether to remove singleton alleles [default TRUE].

`mating` Mating system assumed by the estimator: 'random' or 'monogamy'. The abbreviation 'mono' is also accepted [default 'random'].

`pairing` 'all' [default] if all possible loci should be paired, or 'separate' if only loci on different chromosomes should be used. The chromosome of each locus is read from `x@chromosome`.

`Waples.correction` The type of Waples et al 2016 correction to apply. This is ignored if `pairing` is set to 'separate'. Options are 'nChromosomes', for eq 1a, or 'genomeLength' for eq 1b. NULL if none should be applied [default NULL].

`Waples.correction.value` A single positive number: the number of chromosomes for 'nChromosomes', or the genome length in cM for 'genomeLength'. Required when `Waples.correction` is set. See Waples et al 2016 for details [default NULL].

`naive` Whether the naive (uncorrected for samples size - see eq 7 and eq 8 in Waples 2006) should also be reported. This is mostly to diagnose the source of Inf estimate.

`plot.out` Specify if plot is to be produced [default TRUE].

`plot_theme` User specified theme [default theme_dartR()].

`plot_colors_pop` Either a vector with at least one colour per population, or a palette function taking the number of populations [default gl.select.colors(x)].

`plot.file` Name for the RDS binary files to save (base name only, exclude extension). The plot is saved under this name and the table under the same name with the suffix '_tab'. Nothing is saved when NULL [default NULL].

`plot.dir` Directory in which to save the RDS files [default tempdir(), mandated by CRAN; see gl.check.wd].

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

Value

Invisibly, a named list with one data frame per population. Each data frame holds one row per statistic (lowest allele frequency used, harmonic mean sample size, independent comparisons, overall r^2, expected r^2, estimated Ne and its parametric and jackknife confidence limits) and one column per allele frequency threshold. Five rows are added when `Waples.correction` is set and one when `naive` is TRUE. The full NeEstimator output is written to `file.path(outpath, outfile)`.

Author(s)

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

References

- Waples, R. S. (2006). "A bias correction for estimates of effective population size based on linkage disequilibrium at unlinked gene loci*." Conservation Genetics 7(2): 167-184.
- Waples, R. K., et al. (2016). "Estimating contemporary effective population size in non-model species using linkage disequilibrium across thousands of loci." Heredity 117(4): 233-240.

Examples

```
## Not run:
# SNP data (use two populations and only the first 100 SNPs)
pops <- possums.gl[1:60, 1:100]
nes <- gl.LDNe(pops,
  outfile = "popsLD.txt", outpath = tempdir(),
  neest.path = "./path_to_Ne2-1",
  critical = c(0, 0.05), singleton.rm = TRUE, mating = "random"
)
nes

# Using only pairs of loci on different chromosomes
# make up some chromosome locations
pops@chromosome <- as.factor(sample(1:10, size = nLoc(pops), replace = TRUE))
nessep <- gl.LDNe(pops,
  outfile = "popsLD.txt", outpath = tempdir(), pairing = "separate",
  neest.path = "./path_to_Ne2-1",
  critical = c(0, 0.05), singleton.rm = TRUE, mating = "random"
)
nessep

## End(Not run)
```

`gl.map.popcluster` *Maps a PopCluster plot using a genlight object*

Description

This function takes the Q matrix of a PopCluster run (as returned by `gl.plot.popcluster`) and maps it using the population centres from the genlight object that was used to run PopCluster via `gl.run.popcluster`. It plots the typical structure bar plots on a spatial map, providing a barplot for each subpopulation. Therefore it requires coordinates from a genlight object. This kind of plot should support the interpretation of the spatial structure of a population, but in principle is not different from `gl.plot.popcluster`.

Usage

```
gl.map.popcluster(
  x,
  qmat,
  color_clusters = NULL,
```

```
  provider = "Esri.NatGeoWorldMap",
  scalex = 1,
  scaley = 1,
  movepops = NULL,
  pop.labels = TRUE,
  pop.labels.cex = 12,
  plot.out = TRUE,
  verbose = NULL
)
```

Arguments

x	Name of the genlight object containing the coordinates in the \@other$latlon slot (columns named lon and lat) to calculate the population centres [required].
qmat	Q-matrix returned by gl.plot.popcluster [required].
color_clusters	A colour palette function (for example rainbow), which is called with the number of clusters, or a vector with at least as many colours as clusters [default NULL, which uses gl.select.colors()].
provider	Provider passed to leaflet. Check providers for a list of possible backgrounds [default "Esri.NatGeoWorldMap"].
scalex	Scaling factor to determine the size of the bars in x direction [default 1].
scaley	Scaling factor to determine the size of the bars in y direction [default 1].
movepops	A data frame with two columns, lon and lat, that moves the centre of the barplots manually in case they overlap. Rows are matched to populations by row name when the row names are population names; otherwise there must be one row per population of x, in the order of levels(pop(x)) (see example) [default NULL].
pop.labels	Switch for population labels below the barplots [default TRUE].
pop.labels.cex	Size of population labels [default 12].
plot.out	Specify if the map is to be printed [default TRUE].
verbose	Verbosity: 0, silent or fatal errors; 1, begin and end; 2, brief progress messages; 3, progress and results summary; 5, full report [default 2, unless specified using gl.set.verbosity].

Details

The map is drawn by gl.map.snmf (and so by gl.map.structure), which places each population's bars at its own centre; see there for details. For possible background maps check as specified via the provider: http://leaflet-extras.github.io/leaflet-providers/preview/index.html. You may need to adjust scalex and scaley values [default 1], as the size depends on the scale of the map and the position of the populations.

Value

A list with the map (element map) and the Q matrix split into tables per population, in the order the bars are drawn (element Q_name, with the population in column Pop_name).

Author(s)

Author(s): Ching Ching Lau. Custodian: Ching Ching Lau – Post to https://groups.google.com/d/forum/dartr

References

- Wang, J. (2022). Fast and accurate population admixture inference from genotype data from a few microsatellites to millions of SNPs. Heredity, 129(2), 79-92.

See Also

gl.run.popcluster, gl.plot.popcluster

Other population structure: gl.evanno(), gl.map.snmf(), gl.map.structure(), gl.plot.faststructure(), gl.plot.popcluster(), gl.plot.snmf(), gl.plot.structure(), gl.read.structure(), gl.run.faststructure(), gl.run.popcluster(), gl.run.snmf()

Examples

```
## Not run:
m <- gl.run.popcluster(x = bandicoot.gl,
  popcluster.path = "/User/PopCluster/Bin/", minK = 1, maxK = 3, rep = 2)
Q <- gl.plot.popcluster(pop_cluster_result = m, plot.K = 3, ind_name = TRUE)
gl.map.popcluster(x = bandicoot.gl, qmat = Q)
# move population 4 (out of 5) 0.5 degrees to the right and population 1
# 0.3 degrees to the south of the map.
mp <- data.frame(lon = c(0, 0, 0, 0.5, 0), lat = c(-0.3, 0, 0, 0, 0))
gl.map.popcluster(bandicoot.gl, qmat = Q, movepops = mp)

## End(Not run)
```

---

gl.map.snmf *Maps a snmf plot using a genlight object*

Description

This function takes the Q matrix of a snmf run (as returned by gl.plot.snmf) and maps it using the population centres from the genlight object that was used to run the snmf analysis via gl.run.snmf. It plots the typical snmf bar plots on a spatial map, providing a barplot for each subpopulation. Therefore it requires coordinates from a genlight object. This kind of plot should support the interpretation of the spatial structure of a population, but in principle is not different from gl.plot.snmf.

Usage

```
gl.map.snmf(
  x,
  qmat,
  color_clusters = NULL,
```

```
  provider = "Esri.NatGeoWorldMap",
  scalex = 1,
  scaley = 1,
  movepops = NULL,
  pop.labels = TRUE,
  pop.labels.cex = 12,
  plot.out = TRUE,
  verbose = NULL
)
```

Arguments

x	Name of the genlight object containing the coordinates in the \@other$latlon slot (columns named lon and lat) to calculate the population centres [required].
qmat	Q-matrix returned by gl.plot.snmf (or an element of the matrix list returned by gl.run.snmf) [required].
color_clusters	A colour palette function (for example rainbow), which is called with the number of clusters, or a vector with at least as many colours as clusters [default NULL, which uses gl.select.colors()].
provider	Provider passed to leaflet. Check providers for a list of possible backgrounds [default "Esri.NatGeoWorldMap"].
scalex	Scaling factor to determine the size of the bars in x direction [default 1].
scaley	Scaling factor to determine the size of the bars in y direction [default 1].
movepops	A data frame with two columns, lon and lat, that moves the centre of the barplots manually in case they overlap. Rows are matched to populations by row name when the row names are population names; otherwise there must be one row per population of x, in the order of levels(pop(x)) (see example) [default NULL].
pop.labels	Switch for population labels below the barplots [default TRUE].
pop.labels.cex	Size of population labels [default 12].
plot.out	Specify if the map is to be printed [default TRUE].
verbose	Verbosity: 0, silent or fatal errors; 1, begin and end; 2, brief progress messages; 3, progress and results summary; 5, full report [default 2, unless specified using gl.set.verbosity].

Details

Creates a mapped version of snmf plots with gl.map.structure, which places each population's bars at its own centre. Individuals are matched to x by name (column Label); only the populations present in qmat are mapped. For possible background maps check as specified via the provider: http://leaflet-extras.github.io/leaflet-providers/preview/index.html. You may need to adjust scalex and scaley values [default 1], as the size depends on the scale of the map and the position of the populations.

Value

A list with the map (element map) and the Q matrix split into tables per population, in the order the bars are drawn (element Q_name, with the population in column Pop_name). This can be used to create your own map.

Author(s)

Author(s): Ching Ching Lau. Custodian: Ching Ching Lau – Post to https://groups.google.com/d/forum/dartr

References

- Frichot E, Mathieu F, Trouillon T, Bouchard G, Francois O. (2014). Fast and Efficient Estimation of Individual Ancestry Coefficients. Genetics, 194(4): 973–983.

See Also

gl.run.snmf, gl.plot.snmf, gl.map.structure

Other population structure: gl.evanno(), gl.map.popcluster(), gl.map.structure(), gl.plot.faststructure(), gl.plot.popcluster(), gl.plot.snmf(), gl.plot.structure(), gl.read.structure(), gl.run.faststructure(), gl.run.popcluster(), gl.run.snmf()

Examples

```
## Not run:
m <- gl.run.snmf(x = bandicoot.gl, minK = 1, maxK = 5, rep = 10)
Q <- gl.plot.snmf(snmf.result = m, plot.K = 3, ind.name = TRUE)
gl.map.snmf(bandicoot.gl, qmat = Q)
# move population 4 (out of 5) 0.5 degrees to the right and population 1
# 0.3 degrees to the south of the map.
mp <- data.frame(lon = c(0, 0, 0, 0.5, 0), lat = c(-0.3, 0, 0, 0, 0))
gl.map.snmf(bandicoot.gl, qmat = Q, movepops = mp)

## End(Not run)
```

---

gl.map.structure *Maps a STRUCTURE plot using a genlight object*

Description

This function takes the output of gl.plot.structure (the q matrix) and maps the q-matrix using the population centres from the genlight object that was used to run the structure analysis via gl.run.structure. It plots the typical structure bar plots on a spatial map, providing a barplot for each subpopulation. Therefore it requires coordinates from a genlight object. This kind of plot should support the interpretation of the spatial structure of a population, but in principle is not different from gl.plot.structure.

Usage

```
gl.map.structure(
  qmat,
  x,
  K,
```

```
  provider = "Esri.NatGeoWorldMap",
  scalex = 1,
  scaley = 1,
  movepops = NULL,
  pop.labels = TRUE,
  pop.labels.cex = 12,
  plot.colors = NULL,
  plot.out = TRUE,
  verbose = NULL
)
```

Arguments

qmat	List of q-matrices returned by gl.plot.structure [required].
x	Name of the genlight object containing the coordinates in the \@other$latlon slot (columns named lon and lat) to calculate the population centres [required].
K	The q-matrix to be mapped: either the number of clusters (e.g. 3), which maps the first q-matrix with that many clusters, or a panel label from the K column of the q-matrices (e.g. "2.2" to map the second mode of K = 2) [required].
provider	Provider passed to leaflet. Check providers for a list of possible backgrounds [default "Esri.NatGeoWorldMap"].
scalex	Scaling factor to determine the size of the bars in x direction [default 1].
scaley	Scaling factor to determine the size of the bars in y direction [default 1].
movepops	A data frame with two columns, lon and lat, that moves the centre of the barplots manually in case they overlap (often when populations are horizontally close to each other). Each row gives the longitude and latitude units to move one population. Rows are matched to populations by row name when the row names are population names; otherwise there must be one row per population of x, in the order of levels(pop(x)). Columns not named lon and lat are read as lon, lat (see example) [default NULL].
pop.labels	Switch for population labels below the barplots [default TRUE].
pop.labels.cex	Size of population labels [default 12].
plot.colors	A colour palette function (for example rainbow), which is called with the number of clusters, or a vector with at least as many colours as clusters [default NULL, which uses gl.select.colors(), the default of gl.plot.structure].
plot.out	Specify if the map is to be printed [default TRUE].
verbose	Verbosity: 0, silent or fatal errors; 1, begin and end; 2, brief progress messages; 3, progress and results summary; 5, full report [default 2, unless specified using gl.set.verbosity].

Details

Creates a mapped version of structure plots. For possible background maps check as specified via the provider: http://leaflet-extras.github.io/leaflet-providers/preview/index.html. You may need to adjust scalex and scaley values [default 1], as the size depends on the scale of the map and the position of the populations.

Only the populations present in qmat are mapped; other populations of x are ignored. Each population in qmat needs coordinates in x.

Value

A list with the map (element map) and the q-matrix split into sorted tables per population (element qmats). This can be used to create your own map.

Author(s)

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

References

- Pritchard, J.K., Stephens, M., Donnelly, P. (2000) Inference of population structure using multilocus genotype data. Genetics 155, 945-959.
- Archer, F. I., Adams, P. E. and Schneiders, B. B. (2016) strataG: An R package for manipulating, summarizing and analysing population genetic data. Mol Ecol Resour. doi:10.1111/1755-0998.12559
- Evanno, G., Regnaut, S., and J. Goudet. 2005. Detecting the number of clusters of individuals using the software STRUCTURE: a simulation study. Molecular Ecology 14:2611-2620.
- Mattias Jakobsson and Noah A. Rosenberg. 2007. CLUMPP: a cluster matching and permutation program for dealing with label switching and multimodality in analysis of population structure. Bioinformatics 23(14):1801-1806. Available at clumpp

See Also

gl.run.structure, clumpp, gl.plot.structure

Other population structure: gl.evanno(), gl.map.popcluster(), gl.map.snmf(), gl.plot.faststructure(), gl.plot.popcluster(), gl.plot.snmf(), gl.plot.structure(), gl.read.structure(), gl.run.faststructure(), gl.run.popcluster(), gl.run.snmf()

Examples

```
# examples need structure to be installed on the system (see above)
## Not run:
bc <- bandicoot.gl[,1:100]
sr <- gl.run.structure(bc, k.range = 2:5, num.k.rep = 3, exec = './structure.exe')
ev <- gl.evanno(sr)
ev
qmat <- gl.plot.structure(sr, K = 2:4)
head(qmat)
gl.map.structure(qmat, bc, K = 3)
gl.map.structure(qmat, bc, K = 4)
# move population 4 (out of 5) 0.5 degrees to the right and population 1
# 0.3 degrees to the south of the map.
mp <- data.frame(lon = c(0, 0, 0, 0.5, 0), lat = c(-0.3, 0, 0, 0, 0))
gl.map.structure(qmat, bc, K = 4, movepops = mp)
```

```
## End(Not run)
```

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

Description

This function compares two sets of parental populations to identify loci that exhibit a fixed difference, returns an genlight object with the reduced data, and creates an input file for the program NewHybrids using the top 200 (or hard specified loc.limit) loci. In the absence of two identified parental populations, the script will select a random set 200 loci only (method='random') or the first 200 loci ranked on information content (method='AvgPIC'). If the two parental populations share no fixed difference, 200 loci are selected at random, with a warning.

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. threshold = 0.05 which considers alleles present at greater than 95 a fixed difference. Only the 200 loci are retained, because of limitations of NewHybids.

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

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

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

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

Usage

```
gl.nhybrids(
  gl,
  outpath = tempdir(),
  p0 = NULL,
  p1 = NULL,
  threshold = 0,
  method = "random",
  plot = TRUE,
  plot_theme = theme_dartR(),
  plot_colors = gl.select.colors(ncolors = 2, verbose = 0),
```

```
  pprob = 0.95,
  nhyb.directory = NULL,
  BurnIn = 10000,
  sweeps = 10000,
  GtypFile = "TwoGensGtypFreq.txt",
  AFPriorFile = NULL,
  PiPrior = "Jeffreys",
  ThetaPrior = "Jeffreys",
  verbose = NULL
)
```

Arguments

gl	Name of the genlight object containing the SNP data; SilicoDArT (presence/absence) data are not accepted [required].
outpath	Path where to save the output file [default tempdir()].
p0	List of populations to be regarded as parental population 0 [default NULL].
p1	List of populations to be regarded as parental population 1 [default NULL].
threshold	Sets the level at which a gene frequency difference is considered to be fixed [default 0].
method	Currently has no effect: loci are always selected at random. Retained for back-compatibility [default random].
plot	If TRUE, a plot of the frequency of homozygous reference, heterozygotes and homozygous alternate (SNP) is produced for the F1 individuals [default TRUE, applies only if both parental populations are specified].
plot_theme	User specified theme [default theme_dartR()].
plot_colors	Vector with two color names for the borders and fill [default two colors].
pprob	Threshold level for assignment to likelihood bins [default 0.95, used only if plot=TRUE].
nhyb.directory	Directory that holds the NewHybrids executable file e.g. C:/NewHybsPC. Keep this path short: the NewHybrids executable crashes when the path to its input file (nhyb.directory/nhyb.txt) is 100 or more characters long [default NULL].
BurnIn	Number of sweeps to use in the burn in [default 10000].
sweeps	Number of sweeps to use in computing the actual Monte Carlo averages [default 10000].
GtypFile	Name of a file containing the genotype frequency classes [default TwoGensGtypFreq.txt].
AFPriorFile	Name of the file containing prior allele frequency information [default NULL].
PiPrior	Jeffreys-like priors or Uniform priors for the parameter pi [default Jeffreys].
ThetaPrior	Jeffreys-like priors or Uniform priors for the parameter theta [default Jeffreys].
verbose	Verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log; 3, progress and results summary; 5, full report [default 2 or as specified using gl.set.verbosity].

Value

The reduced genlight object (the loci passed to NewHybrids), not the assignments. The posterior probabilities are written to aa-PofZ.csv in outpath, with columns id, pop, P0, P1, F1, F2, F1xP0 and F1xP1, together with the other NewHybrids output files.

Author(s)

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

References

Anderson, E.C. and Thompson, E.A.(2002). A model-based method for identifying species hybrids using multilocus genetic data. Genetics. 160:1217-1229.

Examples

```
## Not run:
m <- gl.nhybrids(testset.gl,
  p0 = NULL, p1 = NULL,
  nhyb.directory = "D:/workspace/R/NewHybsPC", # Specify as necessary
  outpath = "D:/workspace", # Specify as necessary, usually getwd() [= workspace]
  BurnIn = 100,
  sweeps = 100,
  verbose = 3
)

## End(Not run)
```

---

gl.opt.ind.all *Finds the order of individuals that captures the most alleles*

Description

Ranks individuals so that each one added captures as many new alleles as possible (greedy allele-capture order). Use it to decide how many individuals, and from which populations, are needed to keep a set proportion of the alleles in a sample. Typical uses are choosing founders for a captive breeding program, picking individuals for translocation or population mixing, or setting priorities for sequencing or banking.

Usage

```
gl.opt.ind.all(
  x,
  first = "optimal",
  bootstrap = 100,
  percent = 0.95,
  ncores = 1,
```

```
  plot.display = TRUE,
  plot.theme = theme_dartR(),
  plot.colors = NULL,
  plot.colors.pop = NULL,
  plot.dir = NULL,
  plot.file = NULL,
  verbose = NULL
)
```

Arguments

x	Name of the genlight object containing SNP data [required].
first	How to choose the first individual. 'optimal' starts with the individual carrying the most alleles and runs once; 'random' starts with a random individual and is repeated bootstrap times; 'all' runs once for each individual as the starting individual [default 'optimal'].
bootstrap	Number of runs when first = 'random'. Ignored for 'optimal' and 'all' [default 100].
percent	Proportion of alleles to retain (between 0 and 1). Used to count how many individuals (in total and per population) are needed [default 0.95].
ncores	Number of cores used for 'random' and 'all'. Values above 1 use future::multisession, which works on Windows, Linux and macOS [default 1].
plot.display	Specify if the plot is to be displayed in the graphics window [default TRUE].
plot.theme	User specified theme [default theme_dartR()].
plot.colors	Vector with two colors: the first for lines and borders, the second for fills [default gl.colors(2)].
plot.colors.pop	A color palette for populations or a vector with as many colors as there are populations [default gl.colors("dis")].
plot.dir	Directory to save the plot RDS file [default as specified by the global working directory or tempdir()].
plot.file	Filename (minus extension) for the RDS plot file [required for plot save].
verbose	Verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log; 3, progress and results summary; 5, full report [default 2, unless specified using gl.set.verbosity].

Details

How alleles are counted

At each locus an individual carries one distinct allele if homozygous (0 or 2) and two if heterozygous (1). The function picks the individual that adds the most alleles not yet captured, records them as captured and repeats. A heterozygote captures both alleles at a locus, so the locus is then complete. A homozygote captures one allele; afterwards only individuals that carry the other allele can add anything at that locus. The search stops when all alleles are captured. Individuals that would add no new alleles are therefore not listed in the output.

Missing genotypes (NA) are treated as unknown and add nothing. Individuals with many missing values will look less valuable than they may be, so filter for call rate first (e.g. `gl.filter.callrate(x, method = "ind")`).

Choice of the first individual

The greedy order depends on where it starts. 'optimal' gives one ranking. 'random' and 'all' show how stable the result is when the start changes, and give a spread for the number of individuals needed per population. When several individuals add the same number of alleles, one is picked at random, so use `set.seed()` for reproducible results. Results are reproducible for the same seed whatever the value of `ncores`.

Individuals needed

The individuals needed to reach `percent` are the first k individuals in the order, where k is the first position at which the cumulative proportion of alleles reaches `percent`. This includes the individual that crosses the threshold.

Plot

The upper panel shows the cumulative proportion of alleles captured as individuals are added (one curve per run for 'random' and 'all', with the median in bold). The dashed line marks `percent`. The lower panel shows the individuals needed from each population. For 'optimal' these are bars, with the population size shown in light grey behind. For 'random' and 'all' each bar shows the share of runs in which a population adds 0, 1, 2, ... individuals (classes are grouped when counts are large); the colored part of a bar is the share of runs in which the population is needed at all. Black diamonds show the share of runs that started with an individual from that population.

Starting individual

With 'random' and 'all' the first individual is set by the run, not chosen by the method, and it always counts as needed. When few individuals are needed this can make up a large part of the count and favors populations with many individuals. The plot, `by.run.excl.start` and the summary columns mean_n.excl.start and p.runs therefore leave the starting individual out. With 'optimal' the first individual is chosen by the method and is kept.

If a plot.file is given, the ggplot arising from this function is saved as an "RDS" binary file using saveRDS(); it can be reloaded with readRDS(). If a plot directory (plot.dir) is specified, the ggplot binary is saved to that directory; otherwise to the tempdir().

File-backed (FBM) objects: the algorithm needs the whole genotype matrix in memory. It is decoded with a message stating the memory needed (verbose >= 2); above `getOption("dartR_fbm_max_decode_gb", 4)` Gb the function stops with an error instead of exhausting memory (raise the limit with `options(dartR_fbm_max_decode_gb = ...)` if enough memory is available). See `dartR.base::utils.fbm.decode`.

Value

A list (returned invisibly when `verbose` is 0) with:

- `table` data.frame with one row per individual and run: run, start (id of the first individual), order (position in the order), ind.index (row number in x), id, pop, nall.added (new alleles added), nall.cum (cumulative alleles), prop.cum (cumulative proportion of all alleles) and needed (TRUE if the individual is among those needed to reach `percent`).
- `summary` data.frame with one row per population: n.ind (individuals in x), n.start (runs that started with an individual from this population), the number of individuals needed to reach `percent` as mean_n, median_n, sd, min_n and max_n across runs, prop (share of the total

needed), mean_n.excl.start (mean number needed without the starting individual) and p.runs (share of runs in which the population is needed, without the starting individual). For 'optimal' mean, median, min and max are identical, sd is NA and nothing is excluded.

- `by.run` matrix with one row per run and one column per population: number of individuals needed to reach `percent`.
- `by.run.excl.start` as `by.run`, but without the starting individual of each run (identical to `by.run` for 'optimal').
- `n.needed` vector with the total number of individuals needed in each run.
- `plot` the ggplot (patchwork) object.
- `settings` list of the input settings and data dimensions (first, bootstrap, percent, ncores, n.runs, n.ind, n.loc, n.pop, total.alleles).

Author(s)

Custodian: Bernd Gruber – Post to https://groups.google.com/d/forum/dartr

See Also

gl.filter.callrate

Examples

```
test <- possums.gl[1:60, 1:200]
# one run, starting with the individual with the most alleles
res <- gl.opt.ind.all(test, first = "optimal", percent = 0.95)
res$summary
head(res$table)
# 20 runs, each starting with a random individual
set.seed(42)
res2 <- gl.opt.ind.all(test, first = "random", bootstrap = 20)
res2$summary
# save the table
# write.csv(res2$table, file = "opt.ind.csv", row.names = FALSE)

# small example that runs quickly
res <- gl.opt.ind.all(possums.gl[1:20, 1:50], plot.display = FALSE)
```

---

gl.outflank *Identify Fst outlier loci with the OutFLANK method of Whitlock and Lotterhos (2015)*

Description

Runs OutFLANK on a SNP genlight (or genind) object: a single Fst outlier scan across all populations that flags loci whose Fst is higher (or lower) than expected under a neutral chi-square distribution inferred from the trimmed centre of the Fst distribution.

Usage

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

Arguments

gi	A genlight object with SNP data, or a genind object, with at least two populations defined [required].
plot	Logical: plot the histogram of Fst (uncorrected for sample size) with the inferred neutral distribution [default TRUE].
LeftTrimFraction	The proportion of loci that are trimmed from the lower end of the range of Fst before the likelihood function is applied [default 0.05].
RightTrimFraction	The proportion of loci that are trimmed from the upper end of the range of Fst before the likelihood function is applied [default 0.05].
Hmin	The minimum expected heterozygosity required for a locus to be tested [default 0.1].
qthreshold	The false discovery rate threshold used to flag outliers from the q-values [default 0.05].
verbose	Verbosity: 0, silent or fatal errors; 1, begin and end; 2, brief progress messages; 3, progress and results summary; 5, full report [default 2, unless specified using gl.set.verbosity].

Details

The code of OutFLANK (Whitlock and Lotterhos) is bundled in dartR.popgen, so the OutFLANK package itself is not needed; the Bioconductor package qvalue is. Results are identical to running OutFLANK::MakeDiploidFSTMat() and OutFLANK::OutFLANK() on the same genotypes.

Each SNP enters the analysis once, as the number of copies of the alternate allele (0, 1, 2) per individual. Every input locus is kept in the output, in input order. Loci that are monomorphic or missing in every individual are not tested and get NA. Loci with expected heterozygosity below Hmin are not tested either, but OutFLANK reports them with OutlierFlag FALSE (GoodH = "lowH" in the results), so their index is TRUE.

Negative Fst estimates cannot be tested against the chi-square distribution; they are flagged as outliers only when the lowest positive Fst values are outliers. Low-Fst outliers are not reliable (Whitlock and Lotterhos 2015).

Value

A list with two elements:

- index: a logical vector with one value per input locus, in input order: TRUE for loci that are NOT outliers (including loci below Hmin), FALSE for outliers, NA for monomorphic loci and loci without calls. Outliers are selected with `which(!res$index)`.
- outflank: the OutFLANK output, a list with FSTbar (mean Fst of non-outlier loci), FSTNoCorrbar (the same without sample size correction), dfInferred (inferred degrees of freedom of the neutral chi-square distribution), numberLowFstOutliers, numberHighFstOutliers, and results, a data frame with one row per input locus (LocusName, He, FST, T1, T2, FSTNoCorr, T1NoCorr, T2NoCorr, meanAlleleFreq (frequency of the reference allele), indexOrder, GoodH, qvalues, pvalues, pvaluesRightTail, OutlierFlag).

Author(s)

Author(s): Bernd Gruber; OutFLANK code by Whitlock and Lotterhos. Custodian: Bernd Gruber – Post to https://groups.google.com/d/forum/dartr

References

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

Github repository: Whitlock & Lotterhos: https://github.com/whitlock/OutFLANK

See Also

utils.outflank, utils.outflank.plotter, utils.outflank.MakeDiploidFSTMat

Other selection: gl.run.bayescan()

Examples

```
if (requireNamespace("qvalue", quietly = TRUE)) {
  res <- gl.outflank(bandicoot.gl, plot = FALSE)
  # names of the outlier loci
  locNames(bandicoot.gl)[which(!res$index)]
}
```

---

gl.plot.faststructure *Plots fastStructure analysis results (Q-matrix)*

Description

This function takes a fastStructure run object (output from gl.run.faststructure) and plots the typical structure bar plot that visualises the q matrix of a fastStructure run. The replicate runs are aligned, grouped into modes and averaged exactly as in gl.plot.structure, which draws the plot.

Usage

```
gl.plot.faststructure(
  sr,
  k.range = NULL,
  met_clumpp = "greedyLargeK",
  iter_clumpp = 100,
  clumpak = TRUE,
  plot_theme = NULL,
  colors_clusters = NULL,
  ind_name = TRUE,
  k_name = NULL,
  label.size = 12,
  border_ind = 0.15,
  den = FALSE,
  x = NULL,
  dis.mat = NULL,
  plot.out = TRUE,
  plot.dir = NULL,
  plot.file = NULL,
  verbose = NULL
)
```

Arguments

sr	fastStructure run object from gl.run.faststructure [required].
k.range	The values of K to be plotted. Need to be among the K values in sr. If NULL, all the K's are plotted [default NULL].
met_clumpp	The algorithm to use to infer the correct permutations. One of 'greedy' or 'greedyLargeK' or 'stephens' [default "greedyLargeK"].
iter_clumpp	The number of iterations to use if running either 'greedy' or 'greedyLargeK' [default 100].
clumpak	Whether use the Clumpak method (see details) [default TRUE].
plot_theme	Theme for the plot. See Details for options [default NULL, which uses theme_dartR()].
colors_clusters	A colour palette function (for example rainbow), which is called with the largest K, or a vector with at least as many colours as clusters in the largest K [default NULL, which uses gl.select.colors()].
ind_name	Whether to plot individual names [default TRUE].
k_name	Label of the K panel to plot, as shown in the K column of the returned tables: "3" for K = 3, or "2.1", "2.2", ... when a K has more than one mode. It should be character [default NULL, all panels].
label.size	Size of the population labels [default 12].
border_ind	The width of the border line between individuals [default 0.15].
den	Whether to include a dendrogram. It needs either the genlight object used in gl.run.faststructure (parameter x) or a distance matrix (parameter dis.mat) [default FALSE].

x	The genlight object used in gl.run.faststructure; needed for the dendrogram when dis.mat is not supplied [default NULL].
dis.mat	A dist object (distance matrix) among the individuals, used to build the dendrogram; its labels must be the individual names [default NULL].
plot.out	Specify if plot is to be produced [default TRUE].
plot.dir	Directory in which to save files [default = tempdir(), unless specified using gl.set.wd].
plot.file	Name for the RDS binary file to save (base name only, exclude extension) [default NULL].
verbose	Verbosity: 0, silent or fatal errors; 1, begin and end; 2, brief progress messages; 3, progress and results summary; 5, full report [default 2, unless specified using gl.set.verbosity].

Details

The function outputs a barplot which is the typical output of fastStructure. The fastStructure run object is converted to the layout of a structure run object and passed to gl.plot.structure; see its help for the CLUMPP and Clumpak methods, the averaging of the replicates within each mode, and the dendrogram.

Examples of other themes that can be used can be consulted in

- https://ggplot2.tidyverse.org/reference/ggtheme.html and
- https://yutannihilation.github.io/allYourFigureAreBelongToUs/ggthemes/

Value

A list (invisible) with one data.table per plotted K panel, as returned by gl.plot.structure: one row per individual, with columns Label, cluster1 ... clusterK, K, orig.pop and ord.

Author(s)

Author(s): Bernd Gruber & Luis Mijangos. Custodian: Luis Mijangos – Post to https://groups.google.com/d/forum/dartr

References

- Raj, A., Stephens, M., & Pritchard, J. K. (2014). fastSTRUCTURE: variational inference of population structure in large SNP data sets. Genetics, 197(2), 573-589.
- Kopelman, Naama M., et al. "Clumpak: a program for identifying clustering modes and packaging population structure inferences across K." Molecular ecology resources 15.5 (2015): 1179-1191.
- Mattias Jakobsson and Noah A. Rosenberg. 2007. CLUMPP: a cluster matching and permutation program for dealing with label switching and multimodality in analysis of population structure. Bioinformatics 23(14):1801-1806. Available at clumpp

See Also

gl.run.faststructure, gl.plot.structure

Other population structure: gl.evanno(), gl.map.popcluster(), gl.map.snmf(), gl.map.structure(), gl.plot.popcluster(), gl.plot.snmf(), gl.plot.structure(), gl.read.structure(), gl.run.faststructure(), gl.run.popcluster(), gl.run.snmf()

Examples

```
## Not run:
t1 <- gl.filter.callrate(platypus.gl, threshold = 1)
res <- gl.run.faststructure(t1,
  exec = "./fastStructure", k.range = 2:3,
  num.k.rep = 2, exec.plink = getwd()
)
qmat <- gl.plot.faststructure(res, k.range = 2:3)
gl.map.structure(qmat, K = 2, t1, scalex = 1, scaley = 0.5)

## End(Not run)
```

---

gl.plot.popcluster *Plots PopCluster analysis results (Admixture Model)*

Description

This function takes a PopCluster run object (output from gl.run.popcluster) and plots the typical structure bar plot that visualises the Q matrix of the best run for one K.

Usage

```
gl.plot.popcluster(
  pop_cluster_result,
  border_ind = 0.25,
  plot.K,
  plot_theme = NULL,
  color_clusters = NULL,
  ind_name = T,
  plot.out = TRUE,
  plot.file = NULL,
  plot.dir = NULL,
  verbose = NULL
)
```

Arguments

pop_cluster_result	run object from gl.run.popcluster [required].
border_ind	The width of the border line between individuals [default 0.25].

plot.K	The K of the Q matrix to be plotted: a single value among the K values in the PopCluster run object [required].
plot_theme	Theme for the plot. See Details for options [default NULL].
color_clusters	A colour palette function (for example rainbow), which is called with plot.K, or a vector with at least plot.K colours [default NULL, which uses gl.select.colors()].
ind_name	Whether to plot individual names [default TRUE].
plot.out	Specify if plot is to be produced [default TRUE].
plot.file	Name for the RDS binary file to save (base name only, exclude extension) [default NULL]
plot.dir	Directory in which to save files [default = tempdir(), unless specified using gl.set.wd].
verbose	Verbosity: 0, silent or fatal errors; 1, begin and end; 2, brief progress messages; 3, progress and results summary; 5, full report [default 2, unless specified using gl.set.verbosity].

Details

The function outputs a barplot and Q matrix which is the typical output of PopCluster, for the best run of plot.K chosen by PopCluster (replicates are not averaged). The plot is saved as an RDS file in plot.dir if plot.file is set.

Examples of other themes that can be used can be consulted in

- https://ggplot2.tidyverse.org/reference/ggtheme.html and
- https://yutannihilation.github.io/allYourFigureAreBelongToUs/ggthemes/

The Q matrices can be input to other R packages for plotting ancestry proportion, e.g. FSTruct https://github.com/MaikeMorrison/FSTruct

Value

The Q matrix of plot.K (invisible), as in the matrix element of the gl.run.popcluster result: Index, Order, Label, PercentMiss, Cluster, Pop_1 ... Pop_K, Pop.

Author(s)

Author(s): Ching Ching Lau. Custodian: Ching Ching Lau – Post to https://groups.google.com/d/forum/dartr

References

- Wang, J. (2022). Fast and accurate population admixture inference from genotype data from a few microsatellites to millions of SNPs. Heredity, 129(2), 79-92.

See Also

gl.run.popcluster, gl.map.popcluster

Other population structure: gl.evanno(), gl.map.popcluster(), gl.map.snmf(), gl.map.structure(), gl.plot.faststructure(), gl.plot.snmf(), gl.plot.structure(), gl.read.structure(), gl.run.faststructure(), gl.run.popcluster(), gl.run.snmf()

Examples

```
# examples need popcluster to be installed on the system (see above)
## Not run:
m <- gl.run.popcluster(x = bandicoot.gl,
  popcluster.path = "/User/PopCluster/Bin/", minK = 1, maxK = 3, rep = 10)
Q <- gl.plot.popcluster(pop_cluster_result = m, plot.K = 3, ind_name = TRUE)
gl.map.popcluster(x = bandicoot.gl, qmat = Q)

## End(Not run)
```

`gl.plot.snmf` *Plots ancestry coefficient from snmf*

Description

This function takes a snmf run object (output from `gl.run.snmf`) and plots the typical structure bar plot that visualises the Q matrix of the best run for one K.

Usage

```
gl.plot.snmf(
  snmf.result,
  border.ind = 0.25,
  plot.K,
  plot.theme = NULL,
  color.clusters = NULL,
  ind.name = TRUE,
  plot.out = TRUE,
  plot.file = NULL,
  plot.dir = NULL,
  den = FALSE,
  inverse.den = TRUE,
  x = NULL,
  plot.colors.pop = NULL,
  verbose = NULL
)
```

Arguments

`snmf.result`	run object from `gl.run.snmf` [required].
`border.ind`	The width of the border line between individuals [default 0.25].
`plot.K`	The K of the Q matrix to be plotted: a single value among the K values in the snmf run object [required].
`plot.theme`	Theme for the plot. See Details for options [default NULL].
`color.clusters`	A colour palette function (for example `rainbow`), which is called with plot.K, or a vector with at least plot.K colours [default NULL, which uses gl.colors("structure")].

ind.name	Whether to plot individual names [default TRUE].
plot.out	Specify if plot is to be produced [default TRUE].
plot.file	Name for the RDS binary file to save (base name only, exclude extension) [default NULL].
plot.dir	Directory in which to save files [default = tempdir(), unless specified using gl.set.wd].
den	Whether to include a dendrogram, built by hierarchical clustering (hclust) of Manhattan distances between individuals. It needs the genlight object used in gl.run.snmf in the parameter x [default FALSE].
inverse.den	Flip dendrogram upside down [default TRUE].
x	The genlight object used in gl.run.snmf [default NULL].
plot.colors.pop	A color palette for population plots or a list with as many colors as there are populations in the dataset [default NULL, which uses gl.colors("dis")].
verbose	Verbosity: 0, silent or fatal errors; 1, begin and end; 2, brief progress messages; 3, progress and results summary; 5, full report [default 2, unless specified using gl.set.verbosity].

Details

The function outputs a barplot which is the typical output of snmf. It shows the best run (lowest cross-entropy) for plot.K, as kept by gl.run.snmf; replicates are not averaged. The plot is saved as an RDS file in plot.dir if plot.file is set.

Examples of other themes that can be used can be consulted in

- https://ggplot2.tidyverse.org/reference/ggtheme.html and
- https://yutannihilation.github.io/allYourFigureAreBelongToUs/ggthemes/

The Q matrices can be input to other R packages for plotting ancestry proportion, e.g. FSTruct https://github.com/MaikeMorrison/FSTruct

Value

The Q matrix of plot.K (invisible), as in the matrix element of the gl.run.snmf result: columns Pop_1 ... Pop_K, Cluster, Pop, Label, Order.

Author(s)

Author(s): Ching Ching Lau. Custodian: Ching Ching Lau – Post to https://groups.google.com/d/forum/dartr

References

- Frichot E, Mathieu F, Trouillon T, Bouchard G, Francois O. (2014). Fast and Efficient Estimation of Individual Ancestry Coefficients. Genetics, 194(4): 973–983.

See Also

gl.run.snmf, gl.map.snmf

Other population structure: gl.evanno(), gl.map.popcluster(), gl.map.snmf(), gl.map.structure(), gl.plot.faststructure(), gl.plot.popcluster(), gl.plot.structure(), gl.read.structure(), gl.run.faststructure(), gl.run.popcluster(), gl.run.snmf()

Examples

```
# examples need LEA to be installed on the system (see above)
## Not run:
m <- gl.run.snmf(x = bandicoot.gl, minK = 1, maxK = 5, rep = 10)
Q <- gl.plot.snmf(snmf.result = m, plot.K = 3, ind.name = TRUE)
gl.map.snmf(bandicoot.gl, qmat = Q)

## End(Not run)
```

---

gl.plot.structure *Plots STRUCTURE analysis results (Q-matrix)*

Description

This function takes a structure run object (output from gl.run.structure) and plots the typical structure bar plot that visualises the q matrix of a structure run.

Usage

```
gl.plot.structure(
  sr,
  K = NULL,
  met_clumpp = "greedyLargeK",
  iter_clumpp = 100,
  clumpak = TRUE,
  plot_theme = NULL,
  color_clusters = NULL,
  ind_name = TRUE,
  k_name = NULL,
  border_ind = 0.15,
  den = FALSE,
  dis.mat = NULL,
  x = NULL,
  plot.out = TRUE,
  plot.file = NULL,
  plot.dir = NULL,
  verbose = NULL,
  label.size = 12
)
```

Arguments

sr	Structure run object from gl.run.structure [required].
K	The number for K of the q matrix that should be plotted. Needs to be within the range of K values in your sr structure run object. If NULL, all the K's are plotted [default NULL].
met_clumpp	The algorithm to use to infer the correct permutations. One of 'greedy' or 'greedyLargeK' or 'stephens' [default "greedyLargeK"].
iter_clumpp	The number of iterations to use if running either 'greedy' or 'greedyLargeK' [default 100].
clumpak	Whether use the Clumpak method (see details) [default TRUE].
plot_theme	Theme for the plot. See Details for options [default NULL, which uses theme_dartR()].
color_clusters	A colour palette function (for example rainbow), which is called with the largest K, or a vector with at least as many colours as clusters in the largest K [default NULL, which uses gl.select.colors()].
ind_name	Whether to plot individual names [default TRUE].
k_name	Label of the K panel to plot, as shown in the K column of the returned tables: "3" for K = 3, or "2.1", "2.2", ... when a K has more than one mode. It should be character [default NULL, all panels].
border_ind	The width of the border line between individuals [default 0.15].
den	Whether to include a dendrogram. It needs either the genlight object used in gl.run.structure (parameter x) or a distance matrix (parameter dis.mat) [default FALSE].
dis.mat	A dist object (distance matrix) among the individuals in sr, used to build the dendrogram that orders the structure plot. Its labels must be the individual names. If NULL and den = TRUE, Manhattan distances are calculated from x [default NULL].
x	The genlight object used in gl.run.structure; needed for the dendrogram when dis.mat is not supplied [default NULL].
plot.out	Specify if plot is to be produced [default TRUE].
plot.file	Name for the RDS binary file to save (base name only, exclude extension) [default NULL]
plot.dir	Directory in which to save files [default = tempdir(), unless specified using gl.set.wd].
verbose	Verbosity: 0, silent or fatal errors; 1, begin and end; 2, brief progress messages; 3, progress and results summary; 5, full report [default 2, unless specified using gl.set.verbosity].
label.size	Size of the population labels above the bars [default 12].

Details

The function outputs a barplot which is the typical output of structure. For an Evanno plot use gl.evanno.

This function is based on the methods of CLUMPP and Clumpak as implemented in the R package starmie (https://github.com/sa-lee/starmie).

The Clumpak method identifies sets of highly similar runs among all the replicates of the same K. The method then separates the distinct groups of runs representing distinct modes in the space of possible solutions.

The CLUMPP method permutes the clusters output by independent runs of clustering programs such as structure, so that they match up as closely as possible. The 'greedy' and 'greedyLargeK' searches use random permutations, so cluster order (and colours) can differ between calls unless `set.seed()` is used first.

This function averages the replicates within each mode identified by the Clumpak method.

When den = TRUE, individuals are ordered by a complete-linkage hierarchical clustering (`hclust`) of dis.mat.

The plot is saved as an RDS file in plot.dir if plot.file is set.

Examples of other themes that can be used can be consulted in

- https://ggplot2.tidyverse.org/reference/ggtheme.html and
- https://yutannihilation.github.io/allYourFigureAreBelongToUs/ggthemes/

Value

A list (invisible) with one data.table per plotted K panel, named "1", "2", ... Each table has one row per individual, sorted by Label, with columns Label (individual name), cluster1 ... clusterK (average membership proportions), K (panel label, see k_name), orig.pop (population) and ord (position of the individual in the plot).

Author(s)

Author(s): Bernd Gruber & Luis Mijangos. Custodian: Bernd Gruber – Post to https://groups.google.com/d/forum/dartr

References

- Pritchard, J.K., Stephens, M., Donnelly, P. (2000) Inference of population structure using multilocus genotype data. Genetics 155, 945-959.
- Kopelman, Naama M., et al. "Clumpak: a program for identifying clustering modes and packaging population structure inferences across K." Molecular ecology resources 15.5 (2015): 1179-1191.
- Mattias Jakobsson and Noah A. Rosenberg. 2007. CLUMPP: a cluster matching and permutation program for dealing with label switching and multimodality in analysis of population structure. Bioinformatics 23(14):1801-1806. Available at clumpp

See Also

`gl.run.structure`, `gl.evanno`, `gl.map.structure`

Other population structure: `gl.evanno()`, `gl.map.popcluster()`, `gl.map.snmf()`, `gl.map.structure()`, `gl.plot.faststructure()`, `gl.plot.popcluster()`, `gl.plot.snmf()`, `gl.read.structure()`, `gl.run.faststructure()`, `gl.run.popcluster()`, `gl.run.snmf()`

Examples

```
# examples need structure to be installed on the system (see above)
## Not run:
bc <- bandicoot.gl[,1:100]
sr <- gl.run.structure(bc, k.range = 2:5, num.k.rep = 3, exec = './structure')
ev <- gl.evanno(sr)
ev
qmat <- gl.plot.structure(sr, K=3)
head(qmat)
gl.map.structure(qmat, K=3, bc, scalex=1, scaley=0.5)

## End(Not run)
```

---

gl.read.structure *Read output files produced by the program STRUCTURE*

Description

Reads STRUCTURE output files from a folder and returns them as a structure run object, the same object that `gl.run.structure` returns, so it can be passed to `gl.evanno`, `gl.plot.structure` and `gl.map.structure`. Optionally attaches population and individual names from the genlight object that was analysed.

Usage

```
gl.read.structure(
  folder.path,
  x = NULL,
  pattern = NULL,
  recursive = FALSE,
  rename_files = FALSE,
  prefix = NULL,
  verbose = NULL
)
```

Arguments

`folder.path`	Path to folder containing STRUCTURE output files [required].
`x`	The genlight object that was analysed, used to attach population labels (and individual names) to the q-matrices [default NULL].
`pattern`	Optional regular expression to select files in folder.path (e.g. "_f$") [default NULL].
`recursive`	Logical; search folder recursively [default FALSE].
`rename_files`	Logical; if TRUE, renames the output files on disk to <run name>_out. Stops, before renaming anything, if a file of that name already exists [default FALSE].

prefix	Optional prefix for the run names, e.g. "myrun" gives "myrun.k2.r1" [default NULL, run names "k2.r1"].
verbose	Verbosity: 0, silent or fatal errors; 1, begin and end; 2, brief progress messages; 3, progress and results summary; 5, full report [default 2, unless specified using gl.set.verbosity].

Details

Only STRUCTURE output files are read: files containing both "Estimated Ln Prob of Data" and "Estimated Allele Frequencies". Other files in the folder (data, params, log files) are skipped, with a note at verbose >= 2. K is taken from each file. Within each K, replicates are numbered in the order of the numbers in their file names (so rep2 comes before rep10) and runs are named k<K>.r<replicate>, as in `gl.run.structure`.

Without x, the orig.pop column holds the population number used in the STRUCTURE data file, and id the label from the file. With x, individuals are matched to indNames(x) by name or, when the labels are the numbers 1 to nInd(x) (as in files written by `gl.run.structure`), by position; id then holds the individual names and orig.pop the population names. Labels that match neither way stop the function with an error naming them (STRUCTURE truncates labels longer than 11 characters).

For runs with USEPOPINFO, individuals with a population prior are read like other individuals: their q-matrix row holds the probability of belonging to their given population and, for every other population, the summed probability of ancestry from it over the GENSBACK generations, which is also returned per generation in prior.anc.

Value

A list of class "structure.result", one element per run, named k<K>.r<replicate>. Each element contains:

- summary: named numeric vector (k, est.ln.prob, mean.lnL, var.lnL)
- q.mat: data.frame (id, pct.miss, orig.pop, Group.1..Group.K)
- prior.anc: list of ancestry matrices for individuals with a population prior, or NULL
- files: path of the output file
- label: run name

Author(s)

Author(s): Luis Mijangos & Bernd Gruber. Custodian: Luis Mijangos – Post to https://groups.google.com/d/forum/dartr

See Also

`gl.run.structure`, `gl.evanno`, `gl.plot.structure`

Other population structure: `gl.evanno()`, `gl.map.popcluster()`, `gl.map.snmf()`, `gl.map.structure()`, `gl.plot.faststructure()`, `gl.plot.popcluster()`, `gl.plot.snmf()`, `gl.plot.structure()`, `gl.run.faststructure()`, `gl.run.popcluster()`, `gl.run.snmf()`

Examples

```
## Not run:
# read the output files kept by gl.run.structure
sr <- gl.run.structure(bc, k.range = 2:5, num.k.rep = 3,
                       exec = "./structure", delete.files = FALSE,
                       plot.dir = "structure_runs")
sr2 <- gl.read.structure(list.dirs("structure_runs",
                                   recursive = FALSE)[1], x = bc)
gl.evanno(sr2)
qmat <- gl.plot.structure(sr2, K = 3)

## End(Not run)
```

`gl.run.assignpop` *Runs assignPOP population assignment directly from a genlight object*

Description

Interface between dartRverse genlight objects and the **assignPOP** package. The function prepares the data and, when unknowns are given, runs the assignment.

Single-object mode (`unknown.id = NULL`, `x.unknown = NULL`): `x` contains individuals with known population assignments. The returned object is structurally identical to the output of `assignPOP::read.Structure()` and can be passed directly to `assign.kfold()`, `assign.MC()`, or `assign.matrix()` for cross-validation.

Split mode (`unknown.id` supplied): the simplest mode — a single genlight object containing both reference and unknown individuals. The unknown individuals are identified by name, extracted automatically, and the remainder becomes the reference set. Reference populations with fewer than `nmin` individuals are dropped. Population assignment is performed by `assignPOP::assign.X()` internally.

Two-object mode (`x.unknown` supplied): `x` contains the reference individuals (knowns) and `x.unknown` contains individuals whose population of origin is to be determined. Reference populations with fewer than `nmin` individuals are dropped. Population assignment is performed by `assignPOP::assign.X()` internally.

In all two-dataset modes the two objects are reduced to their common set of loci before encoding.

Internally, genotype dosage values (0, 1, 2) are one-hot encoded into allele frequency pairs per locus, replicating the `structure_onehot()` step performed by `read.Structure()`:

- dosage 0 (homozygous reference) → `[1.0, 0.0]`
- dosage 1 (heterozygous) → `[0.5, 0.5]`
- dosage 2 (homozygous alternate) → `[0.0, 1.0]`
- `NA` (missing) → `[0.0, 0.0]`

Each locus contributes two columns to `DataMatrix`, named `<locusname>_1` and `<locusname>_2`.

Usage

```
gl.run.assignpop(
  x,
  unknown.id = NULL,
  x.unknown = NULL,
  nmin = 10,
  dir = tempdir(),
  model = "svm",
  svm.kernel = "linear",
  svm.cost = 1,
  ntree = 50,
  pca.PCs = "kaiser-guttman",
  verbose = NULL
)
```

Arguments

x	Genlight object. In single-object and split modes this is the only object required. In two-object mode it contains the known reference individuals [required].
unknown.id	Character vector of individual names within x that are to be treated as unknowns. These are extracted from x automatically; the remainder of x becomes the reference set. Cannot be used together with x.unknown [default NULL].
x.unknown	Genlight object containing individuals whose population of origin is unknown. Population labels in this object are ignored. Cannot be used together with unknown.id [default NULL].
nmin	Minimum number of individuals required in a reference population for it to be retained. Populations with fewer individuals are dropped with a warning. Applied only when unknowns are present [default 10].
dir	Directory where assignPOP::assign.X() writes its output files (including AssignmentResult.txt). Only used in split and two-object modes [default tempdir()].
model	Classification model passed to assignPOP::assign.X(). One of "svm", "lda", "naiveBayes", "tree", or "randomForest" [default "svm"].
svm.kernel	Kernel type for the SVM classifier, passed to assignPOP::assign.X(). One of "linear", "radial", "polynomial", or "sigmoid" [default "linear"].
svm.cost	Cost parameter for the SVM classifier, passed to assignPOP::assign.X() [default 1].
ntree	Number of trees for the random forest classifier, passed to assignPOP::assign.X() [default 50].
pca.PCs	Number of principal components to retain for dimensionality reduction inside assign.X(), or "kaiser-guttman" to select automatically [default "kaiser-guttman"].
verbose	Verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log; 3, progress and results summary; 5, full report [default 2, unless specified using gl.set.verbosity].

Value

Single-object mode: a named 3-element list identical in structure to the output of `assignPOP::read.Structure()`:

`DataMatrix` A data frame of one-hot encoded allele frequencies with a final `popNames_vector` column.

`SampleID` Character vector of individual identifiers.

`LocusName` Character vector of locus names.

Split mode or two-object mode: a named list with three elements:

`results` A data frame containing the assignment result for each unknown individual: columns `Ind.ID`, `pred.pop`, and one column per reference population giving the posterior probability. Also written to `AssignmentResult.txt` in `dir`.

`train` The assignPOP data object for the reference individuals, suitable for subsequent cross-validation with `assign.kfold()` or `assign.MC()`.

`unknowns` The assignPOP data object for the unknown individuals.

Author(s)

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

Examples

```
require("dartR.data")
if (isTRUE(getOption("dartR_fbm"))) possums.gl <- gl.gen2fbm(possums.gl)

# Single-object mode: cross-validation on known individuals
dat <- gl.run.assignpop(possums.gl, verbose = 3)
# assignPOP::assign.kfold(dat, k.fold = 5, dir = tempdir())

# Split mode: unknown extracted by name, assignment run automatically
# out <- gl.run.assignpop(possums.gl, unknown.id = "1", nmin = 10,
#                         verbose = 3)
# out$results   # assignment result data frame
# out$train     # reference data object
# out$unknowns  # unknown data object
```

---

gl.run.bayescan *Identify Fst outlier loci with BayeScan*

Description

Runs BayeScan (Foll and Gaggiotti 2008) on a SNP genlight object to identify loci under selection, and returns one row of results per locus with its classification as neutral, under diversifying selection or under balancing selection. BayeScan is not distributed with dartR.popgen: the user must download it from https://cmpg.unibe.ch/software/BayeScan/ and give its location in `bayescan.path`. Please cite the original publication if you use this function.

Usage

```
gl.run.bayescan(
  x,
  bayescan.path,
  prior.odds = 10,
  n.iter = 5000,
  thin = 10,
  n.pilot = 20,
  pilot.length = 5000,
  burnin = 50000,
  n.cores = NULL,
  q.threshold = 0.05,
  outfile = "bayescan",
  outpath = NULL,
  plot.display = TRUE,
  plot.theme = theme_dartR(),
  plot.colors = c("grey60", "deeppink", "dodgerblue"),
  plot.dir = NULL,
  plot.file = NULL,
  verbose = NULL
)
```

Arguments

x	Name of the genlight object containing SNP data, with at least two populations defined [required].
bayescan.path	Path to the folder holding the BayeScan executable, or to the executable itself. In a folder, the function looks for the file names of the BayeScan 2.1 distribution for the operating system (e.g. BayeScan2.1_macos64bits) and for "bayescan" [required].
prior.odds	Prior odds for the neutral model. Option -pr_odds in BayeScan [default 10].
n.iter	Number of MCMC iterations kept for the posterior. Option -n in BayeScan [default 5000].
thin	Thinning interval: number of iterations between two kept iterations. Option -thin in BayeScan [default 10].
n.pilot	Number of pilot runs used to tune the proposal distributions. Option -nbp in BayeScan [default 20].
pilot.length	Length of each pilot run. Option -pilot in BayeScan [default 5000].
burnin	Number of burn-in iterations. Option -burn in BayeScan [default 50000].
n.cores	Number of threads used by BayeScan. If NULL, BayeScan uses all available CPUs. Option -threads in BayeScan [default NULL].
q.threshold	Largest q-value (false discovery rate) for a locus to be classified as under selection [default 0.05].
outfile	Prefix of the names of the input and output files of BayeScan [default "bayescan"].

outpath	Path where to save the input and output files of BayeScan [default global working directory or if not specified, tempdir()].
plot.display	Specify if the plot of Fst against log10(q-value) and the plot of the MCMC trace are to be displayed in the graphics window [default TRUE].
plot.theme	User specified theme [default theme_dartR()].
plot.colors	Vector with three colour names for neutral loci, loci under diversifying selection and loci under balancing selection [default c("grey60", "deeppink", "dodgerblue")].
plot.dir	Directory to save the plot RDS files [default as specified by the global working directory or tempdir()].
plot.file	Filename (minus extension) for the RDS plot file; the trace plot is saved with "_trace" added to the name [Required for plot save].
verbose	Verbosity: 0, silent or fatal errors; 1, begin and end; 2, brief progress messages; 3, progress and results summary; 5, full report [default 2, unless specified using gl.set.verbosity].

Details

BayeScan splits the Fst of each locus in each population into a locus effect (alpha), shared by all populations, and a population effect (beta), shared by all loci. A locus needs a non-zero alpha to explain its allele frequencies when it is under selection: a positive alpha means more differentiation than expected (diversifying selection), a negative alpha less (balancing selection). BayeScan estimates by reversible-jump MCMC the posterior probability that alpha is non-zero, and from it a q-value, the smallest false discovery rate at which the locus would be called an outlier. Here a locus is classified as under selection when its q-value is at most `q.threshold`, and as diversifying or balancing by the sign of alpha.

`prior.odds` sets how much more likely the neutral model is than the model with selection before the data are seen. The BayeScan default of 10 gives many false positives when thousands of loci are tested, or when the populations have a hierarchical structure or a history of range expansion (Lotterhos and Whitlock 2014); larger values, e.g. 100, make the test more conservative.

Each population in `pop(x)` is a sample in the analysis, so subset `x` to the populations to be compared (`gl.keep.pop`) before running the function. Loci that are monomorphic in `x`, or have no calls, carry no information on differentiation; they are not sent to BayeScan and get NA in the results. A population with no calls at a locus is passed to BayeScan as a sample of zero gene copies.

BayeScan has no option to set the random seed, so results vary slightly between runs. Check that the chain has converged before using the results. In the trace plot, each parameter should vary around a stable level, with no trend and no long excursions. When the package coda is installed, the effective sample size (ESS: the number of independent draws the autocorrelated trace is worth) of each parameter is shown in the panel titles and given for each population in `fst.pop`, and the function warns when the ESS of the Fst of a population is below 200. The log-likelihood usually has a much lower ESS than the population Fst (137 against more than 3000 on the BayeScan example data with the default settings). The trace covers the log-likelihood and the Fst of each population, not the alpha of each locus. If the chain has not converged, increase `burnin` and `n.iter`. Runs on tens of thousands of loci take hours.

The input file and all BayeScan output files are written to `outpath`, with names starting with `outfile`: the input file (_input.txt), the results per locus (_fst.txt), the MCMC trace of the log-likelihood and the Fst of each population (.sel), the acceptance rates (_AccRte.txt), a summary of

the run settings (_Verif.txt) and the screen output of BayeScan (.log). Files from an earlier run with the same `outfile` and `outpath` are overwritten.

Value

A list with five elements:

- results: a data frame with one row per input locus, in input order: locus (locus name), prob (posterior probability of the model with selection), log10.PO (log10 of the posterior odds for the model with selection; BayeScan writes 1000 when prob is 1), q.value, alpha (locus effect), fst (Fst of the locus averaged over populations) and selection (neutral, diversifying or balancing). Loci not tested are NA.
- fst.pop: a data frame with the posterior mean and the 2.5% and 97.5% quantiles of the Fst of each population, and the effective sample size of its trace (ess; NA when the package coda is not installed).
- trace: a data frame with the MCMC trace of the log-likelihood (logL) and of the Fst of each population, one row per kept iteration.
- plot: a ggplot of Fst against log10(q-value), with a dashed line at `q.threshold`. q-values below 0.0001 are drawn at 0.0001.
- trace.plot: a ggplot of the MCMC trace, one panel for the log-likelihood and one for the Fst of each population, against the iteration; the panel titles give the effective sample size when the package coda is installed.

Author(s)

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

References

- Foll, M. and Gaggiotti, O. (2008) A genome-scan method to identify selected loci appropriate for both dominant and codominant markers: a Bayesian perspective. Genetics 180: 977-993. doi:10.1534/genetics.108.092221
- Lotterhos, K.E. and Whitlock, M.C. (2014) Evaluation of demographic history and neutral parameterization on the performance of FST outlier tests. Molecular Ecology 23: 2178-2192. doi:10.1111/mec.12725

See Also

`gl.outflank`, `gl2bayescan`

Other selection: `gl.outflank()`

Examples

```
## Not run:
# BayeScan downloaded from https://cmpg.unibe.ch/software/BayeScan/
x <- gl.keep.pop(possums.gl, pop.list = c("A", "B"))
res <- gl.run.bayescan(x, bayescan.path = "path/to/BayeScan2.1/binaries",
```

```
                        prior.odds = 100)
# loci under diversifying selection
res$results[which(res$results$selection == "diversifying"), ]

## End(Not run)
```

---

gl.run.epos *Run EPOS for inference of historical population-size changes*

Description

This function runs EPOS (based on Lynch et al. 2020) to estimate historical population-size https://github.com/EvolBioInf/epos. It relies on a compiled version of the software epos, epos2plot and if a bootstrap output is required bootSfs. For more information on the approach check the publication (Lynch et al. 2020), the github repository https://github.com/EvolBioInf/epos and look out for the manual epos.pdf (https://github.com/EvolBioInf/epos/blob/master/doc/epos.pdf). The binaries need to be provided in a single folder and can be downloaded via the gl.download.binary function (including the necessary dlls for windows; under Linux gsl, blas need to be installed on your system). Please note: if you use this method, make sure you cite the original publication in your work.

Usage

```
gl.run.epos(
  x,
  epos.path,
  sfs = NULL,
  minbinsize = 1,
  folded = TRUE,
  L = NULL,
  u = NULL,
  boot = 0,
  upper = 0.975,
  lower = 0.025,
  method = "greedy",
  depth = 2,
  other.options = "",
  seed = NULL,
  outfile = "epos.out",
  outpath = NULL,
  cleanup = TRUE,
  plot.display = TRUE,
  plot.theme = theme_dartR(),
  plot.dir = NULL,
  plot.file = NULL,
  verbose = NULL
)
```

Arguments

x	dartR/genlight object with SNP data [required].
epos.path	path to the folder with epos and epos2plot (always required) and bootSfs (required if boot > 0) [required].
sfs	the SFS to use instead of calculating it from x. Either named as returned by gl.sfs ("d0", "d1", ...; the names give the classes), or an unnamed vector of counts for the classes minbinsize, minbinsize + 1, ... An unfolded SFS (folded = FALSE) of n individuals has the classes 1 to 2n-1; a last class 2n (loci fixed for the alternative allele, which gl.sfs returns) is dropped [default NULL, calculated with gl.sfs(x, singlepop = TRUE)].
minbinsize	the smallest SFS class used. 0 keeps the zero class (monomorphic sites), which EPOS then uses instead of L; 1 (default) drops it; 2 also excludes singletons, and so on (option -x in epos). If your genlight object has more than one population the sfs is calculated with singlepop set to TRUE (one sfs for all individuals), as epos does not work with a multidimensional sfs [default 1].
folded	if set to TRUE (default) a folded sfs (minor allele frequency sfs) is used. If set to FALSE an unfolded (derived allele frequency sfs) is used. It is assumed that 0 is homozygote for the reference and 2 is homozygote for the derived allele. So you need to make sure your coding is correct. Option -U in epos [default TRUE].
L	length of sequences (including monomorphic and polymorphic sites) for all the loci in x. Required unless minbinsize = 0. If loci with missing calls are excluded from the SFS, L is scaled to the loci used (see details). Option -l in epos [default NULL].
u	mutation rate. If not provided the default value of epos is used (5e-9). Option -u in epos [default NULL].
boot	if set to a value >0 the program bootSfs is used to provide multiple bootstrapped sfs, which allows to calculate confidence intervals of the historic Ne sizes. Be aware the runtime can be extended. 0: no bootstrapped simulations are run, otherwise boot number of bootstraps are run (option -i in bootSfs) [default 0].
upper	upper quantile of the bootstrap (only used if boot>0). Option -u in epos2plot [default 0.975].
lower	lower quantile of the bootstrap (only used if boot>0). Option -l in epos2plot [default 0.025].
method	either "exhaustive" or "greedy". Under "greedy" each round of the search adds one level (a change in population size) to the levels already found. Under "exhaustive" every combination of levels is compared in each round up to round depth; later rounds are greedy. The first two rounds are the same under both methods, so "exhaustive" differs from "greedy" only if depth is 3 or more. Check the epos manual for details [default "greedy"].
depth	the number of rounds searched exhaustively if method = "exhaustive" (option -E in epos); ignored if method = "greedy". Values below 3 give the greedy search. The run time rises steeply with depth and with the number of individuals, and a deeper search does not always end with a higher likelihood, because the greedy rounds that follow start from a different set of levels [default 2].

other.options	additional options for epos, as typed on the command line (e.g "-m 10") [default ""].
seed	seed for the random number generators of epos and bootSfs (option -s). If NULL, both programs use a seed from the system clock and results differ between runs [default NULL].
outfile	File name of the output file (the raw epos output) [default 'epos.out'].
outpath	Path where to save the output file [default global working directory or if not specified, tempdir()].
cleanup	if set to true intermediate tempfiles are deleted after the run [default TRUE].
plot.display	Specify if plot is to be produced [default TRUE].
plot.theme	User specified theme [default theme_dartR()].
plot.dir	Directory to save the plot RDS files [default as specified by the global working directory or tempdir()]
plot.file	Filename (minus extension) for the RDS plot file [Required for plot save]
verbose	Verbosity: 0, silent or fatal errors; 1, begin and end; 2, brief progress messages; 3, progress and results summary, including the messages of epos; 5, full report [default 2, unless specified using gl.set.verbosity].

Details

EPOS (Estimation of Population Size changes) is a software tool that infers historical changes in population size using allele-frequency data from population-genomic surveys. The method relies on the site-frequency spectrum (SFS) of nearly neutral polymorphisms. The underlying theory uses coalescence models, which describe how gene sequences have originated from a common ancestor. By analyzing the probability distributions of the starting and ending points of branch segments over all possible coalescence trees, EPOS can estimate historic population sizes.

The function uses a model-flexible approach, meaning it estimates historic population sizes, without the necessity to provide a candidate scenario. An efficient statistical procedure is employed, to estimate historic effective population sizes.

For all the possible settings, please refer to the manual of EPOS.

The main parameters that are necessary to run the function are a genlight/dartR object, L (length of sequences), u (mutation rate), and the path to the epos binaries. For details check the example below.

Please note: There is currently not really a good way to estimate L, the length of all sequences. Often users of dart data use the number of loci multiplied by 69, but this is definitely an underestimate as monomorphic loci need to be included (also the length of the restriction site should be added for each loci). For mutation rate u, the default value is set to 5e-9, but should be adapted to the species of interest. The good news is, that settings of L and mu affects only the axis of the inferred history, but not the shape of the history. So users can infer the shape, but need to be careful with a temporal interpretation as both x and y axis are affected by the mutation rate and L.

EPOS needs the number of monomorphic sites, either as the sequence length L (option -l) or as the zero class of the SFS (`minbinsize = 0`). The zero class counts only the monomorphic loci kept in the genlight object, which for DArT data is usually far below the true number of monomorphic sites, so L is usually the better choice.

The SFS is sent to EPOS with its true class numbers: classes below `minbinsize` are excluded with the EPOS option -x, and an unfolded SFS (`folded = FALSE`) is sent as classes 1 to 2n-1 with option

-U.
The SFS is built only from the loci scored in every individual, because a locus with missing calls cannot be placed in a frequency class of the full sample (see `gl.sfs`). `L` is taken to be the sequence length of all the loci in `x`: when loci are excluded for missing calls, EPOS is given `L` multiplied by the proportion of loci used, and the number of loci excluded and the value of L used are reported (verbose >= 1). Without this correction Ne and the times would be too low by that proportion. To keep more loci, remove individuals with low call rate or impute (`gl.impute`) before running the analysis. With a user-supplied `sfs`, `L` is used as given.
EPOS assumes that the sample comes from one population without substantial structure (Lynch et al. 2020). If `x` holds several populations they are pooled into one SFS, and pooling differentiated populations distorts the SFS and the history inferred from it. Run each population on its own unless the populations in `x` form one genetic population.
EPOS writes a population size of 1 where its estimate for an interval is negative. An Ne of 1 in the history is therefore not an estimate: it marks an interval, often the most recent one, whose size EPOS could not estimate. These intervals are kept in the returned history and in the plot, and the function warns about them (verbose >= 1).

Value

returns a list with four components:

- history: Ne estimates over generations (generation, low, median and high)
- plot: a ggplot of history
- sfs: the sfs used for the analysis (class r and count fr)
- diagnostics: a list with the several diagnostics and a plot of observed and expected sfs

Author(s)

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

References

Lynch, Michael, Bernhard Haubold, Peter Pfaffelhuber, and Takahiro Maruki. 2020. Inference of Historical Population-Size Changes with Allele-Frequency Data. G3: Genes|Genomes|Genetics 10, no. 1: 211-23. doi:10.1534/g3.119.400854.

See Also

Other demographic history: `gl.TajimasD()`, `gl.run.stairway2()`, `gl.sfs()`

Examples

```
## Not run:
#gl.download.binary("epos",os="windows")
require(dartR.data)
# one population: the first 30 individuals of possums.gl
epos <- gl.run.epos(possums.gl[1:30, ], epos.path = file.path(tempdir(),"epos"),
                    L=1e5, u = 1e-8)
epos$history
```

```
## End(Not run)
```

`gl.run.faststructure` *Runs a faststructure analysis using a genlight object*

Description

This function takes a genlight object and runs a faststructure analysis.

Usage

```
gl.run.faststructure(
  x,
  k.range,
  num.k.rep = 1,
  exec = "./fastStructure",
  exec.plink = getwd(),
  output = tempdir(),
  tol = 1e-05,
  prior = "simple",
  cv = 0,
  seed = NULL,
  verbose = NULL,
  plot.out = TRUE,
  plot.theme = theme_dartR(),
  plot.dir = NULL,
  plot.file = NULL
)
```

Arguments

`x`	Name of the genlight object containing the SNP data [required].
`k.range`	Values of K (number of populations) to run, e.g. 2:5 or c(1, 3, 5) [required].
`num.k.rep`	Number of replicates for each K [default 1].
`exec`	Full path and name of the fastStructure executable [default "./fastStructure"].
`exec.plink`	Path of the folder that holds the PLINK executable, named plink [default getwd()].
`output`	Folder in which a new subfolder is created for the files of this run (PLINK input, fastStructure output) [default tempdir()].
`tol`	Convergence criterion [default 10e-6].
`prior`	Choice of prior: simple or logistic [default "simple"].
`cv`	Number of test sets for cross-validation, 0 implies no CV step [default 0].

seed	Seed for the random number generator; replicate r uses seed + r - 1, so replicates differ and can be repeated [default NULL].
verbose	Verbosity: 0, silent or fatal errors; 1, begin and end; 2, one line per run; 3, progress and results summary, including the output of PLINK and fastStructure; 5, full report [default 2, unless specified using gl.set.verbosity].
plot.out	Specify if the marginal likelihood plot is to be produced [default TRUE].
plot.theme	Theme for the plot [default theme_dartR()].
plot.dir	Directory in which to save the plot [default = tempdir(), unless specified using gl.set.wd].
plot.file	Name for the RDS binary file to save the plot (base name only, exclude extension) [default NULL].

Details

fastStructure does not run on Windows. Download the faststructure binary for Mac or Linux from here:

https://github.com/StuntsPT/Structure_threader/tree/master/structure_threader/bins

and make it executable, e.g. `system(paste0("chmod u+x ", getwd(), "/fastStructure"))`.

Download the PLINK 1.9 binary for your system from here:

https://www.cog-genomics.org/plink/

and pass the folder that holds it as exec.plink. The genlight object is converted to PLINK binary files with gl2plink.

To install fastStructure dependencies follow these directions: https://github.com/rajanil/fastStructure

fastStructure performs inference for the simplest, independent-loci, admixture model, with two choices of priors that can be specified using the –prior parameter. Thus, unlike Structure, fastStructure does not require the mainparams and extraparam files. The inference algorithm used by fastStructure is fundamentally different from that of Structure and requires the setting of far fewer options.

To identify the number of populations that best approximates the marginal likelihood of the data, the marginal likelihood is extracted from each run of K, averaged across replicates and plotted.

Each call writes its files to a new subfolder of output (named fastStructure_<date>_<time>), so files from earlier runs are never read.

Value

A list with two elements: q_list, a list named by K, each element a list named by replicate ("1", "2", ...) of data frames with one row per individual (columns id, orig.pop, V1 ... VK: the fastStructure q-matrix); and plot, the ggplot of the mean marginal likelihood per K.

Author(s)

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

References

- Raj, A., Stephens, M., & Pritchard, J. K. (2014). fastSTRUCTURE: variational inference of population structure in large SNP data sets. Genetics, 197(2), 573-589.

See Also

gl.plot.faststructure

Other population structure: gl.evanno(), gl.map.popcluster(), gl.map.snmf(), gl.map.structure(), gl.plot.faststructure(), gl.plot.popcluster(), gl.plot.snmf(), gl.plot.structure(), gl.read.structure(), gl.run.popcluster(), gl.run.snmf()

Examples

```
## Not run:
# Please note: faststructure needs to be installed
# Please note: faststructure is not available for windows
t1 <- gl.filter.callrate(platypus.gl, threshold = 1)
res <- gl.run.faststructure(t1,
  exec = "./fastStructure", k.range = 2:3,
  num.k.rep = 2, exec.plink = getwd()
)
qmat <- gl.plot.faststructure(res, k.range = 2:3)
gl.map.structure(qmat, K = 2, t1, scalex = 1, scaley = 0.5)

## End(Not run)
```

---

gl.run.mongrail *Classify hybrids with MONGRAIL from a genlight object*

Description

Runs MONGRAIL, a Bayesian full-likelihood method that classifies individuals into one of six two-generation genealogical classes – pure population A, pure population B, F1, F2, backcross to A, and backcross to B – from multilocus SNP genotypes of two reference populations plus a set of test individuals. The function writes the three MONGRAIL `.GT` input files from a genlight object, invokes the `mongrail2` executable, and returns the posterior probability of each class for every test individual together with the most probable class.

Usage

```
gl.run.mongrail(
  x,
  pop.A,
  pop.B,
  test.pop = NULL,
  mongrail.path = NULL,
  outpath = NULL,
```

```
  output.name = "mongrail",
  recomb.rate = 1e-08,
  max.snps.per.chr = 16,
  prune.threshold = NULL,
  max.recomb = NULL,
  phased = FALSE,
  verbose = NULL
)
```

Arguments

x	Name of the genlight object [required].
pop.A	Population name(s) in pop(x) forming reference population A [required].
pop.B	Population name(s) in pop(x) forming reference population B [required].
test.pop	Population name(s) in pop(x) to classify. NULL uses all individuals not assigned to pop.A or pop.B [default NULL].
mongrail.path	Directory holding the mongrail2 executable [default getOption("mongrail.path")].
outpath	Directory for the .GT input files and the output [default tempdir()].
output.name	Base name for the output file [default "mongrail"].
recomb.rate	Recombination rate per base pair, passed as -r [default 1e-8].
max.snps.per.chr	Maximum SNPs retained per chromosome; chromosomes with more are thinned to an evenly spaced subset. The MONGRAIL .GT hard limit is 32, and exact enumeration is used at or below 16 [default 16].
prune.threshold	Posterior probability threshold for pruning haplotype pairs, passed as -p (0-1). NULL omits it [default NULL].
max.recomb	Maximum recombinations for sparse enumeration, passed as -k. Required when any chromosome holds more than 16 loci; NULL lets the function set it when needed [default NULL].
phased	If TRUE, heterozygotes are written phased (a|b); if FALSE they are written unphased (a/b), which suits DArTseq data [default FALSE].
verbose	Verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log; 3, progress and results summary; 5, full report [default NULL, unless specified using gl.set.verbosity].

Details

MONGRAIL models linkage and recombination along chromosomes, so loci are grouped by their chromosome and position (from x@chromosome and x@position). When those slots are empty – as for an anonymous SNP panel with no reference genome – every locus is placed on its own chromosome, i.e. treated as unlinked, which is the correct default in the absence of map information. Because the method scales steeply with the number of loci per chromosome, panels mapped to a genome are thinned to max.snps.per.chr loci per chromosome.

The .GT format tolerates unphased genotypes, so the two-allele DArTseq calls are written directly (0/0, 0/1, 1/1) with missing genotypes as ./. and no phasing step is required.

MONGRAIL and its mongrail2 executable are the work of the MONGRAIL authors; this function only prepares its input and parses its output. See https://github.com/mongrail/mongrail2.

Value

A list with $assignments (a data.frame of test individual, most probable class, and the six posterior probabilities), $files (the input and output file paths) and $raw (the parsed output table), returned invisibly.

Author(s)

Custodian: Peter J. Unmack – Post to https://groups.google.com/d/forum/dartr

Examples

```
## Not run:
res <- gl.run.mongrail(testset.gl, pop.A = "pop1", pop.B = "pop2",
                       mongrail.path = getOption("mongrail.path"))
head(res$assignments)

## End(Not run)
```

---

gl.run.popcluster *Runs a PopCluster analysis using a genlight object*

Description

Creates an input file for the program PopCluster and runs it if PopCluster is installed (can be installed at: https://www.zsl.org/about-zsl/resources/software/popcluster)

If you specify a directory for the PopCluster executable file, then the script will create the input file (DataForm=0) from the SNP data then run PopCluster.

PopCluster infers population admixture by coupling a clustering stage with a subsequent admixture-analysis stage. First, it uses simulated annealing to assign individuals to clusters under a mixture model, thus identifying discrete populations and estimating allele frequencies without prematurely converging to local optima. In the second step, these results provide starting points for an expectation–maximization (EM) algorithm under an admixture model, where each individual's genetic contributions from multiple populations are refined.

Refer to the PopCluster manual for further information on the parameters to set.

Usage

```
gl.run.popcluster(
  x,
  popcluster.path = getwd(),
  output.path = tempdir(),
  filename = "output",
  minK = 1,
  maxK = 2,
  rep = 1,
```

```
  Scaling = 0,
  search_relate = 0,
  allele_freq = 1,
  ISeed = 333,
  PopFlag = 0,
  model = 2,
  loc_admixture = 0,
  relatedness = 0,
  kinship = 0,
  pr_allele_freq = 2,
  parallel = FALSE,
  ncores = 1,
  cleanup = TRUE,
  plot.dir = NULL,
  plot.out = TRUE,
  plot.file = NULL,
  plot_theme = theme_dartR(),
  verbose = NULL
)
```

Arguments

x	Name of the genlight object containing the SNP data [required].
popcluster.path	Path to the directory that contains the PopCluster program (PopClusterMac, PopClusterLnx or PopClusterWin.exe) [default getwd()].
output.path	Folder in which the PopCluster parameter file and input data are written; created if missing [default tempdir()].
filename	Prefix of all the files that will be produced [default "output"].
minK	Minimum K [default 1].
maxK	Maximum K [default 2].
rep	Number of replicates runs per K [default 1].
Scaling	Scaling to be applied in the clustering analysis: none (0), weak (1), medium (2), strong (3) and very strong (4), see details section [default 0].
search_relate	Method for proposing a configuration in clustering analysis. 0 for the assignment probability method and 1 for relatedness method. [default 0].
allele_freq	Output allele frequency: 0=N, 1=Y [default 1].
ISeed	Seed for random number generator [default 333].
PopFlag	Whether to use population information stored in the genlight object in the slot "pop" in structure analysis. 0=No and 1=Yes [default 0].
model	1=Clustering, 2=Admixture, 3=Hybridyzation, 4=Migration model [default 2].
loc_admixture	Whether to estimate and output the admixture proportions for each individual at each locus (=1) or not (=0) [default 0].
relatedness	Compute relatedness = 0=No, 1=Wang, 2=LynchRitland [default 0].

kinship	Estimate kinship: 0=N, 1=Y [default 0].
pr_allele_freq	Whether allele frequency prior should be determined by the program (0), the Equal Frequency prior (1) or Unequal Frequency prior (2) [default 2].
parallel	Use parallelisation (implemented only in LINUX for the moment) [default FALSE].
ncores	How many cores should be used [default 1].
cleanup	Remove the temporary folder in which PopCluster runs (a copy of the program and all its output files) when finished [default TRUE].
plot.dir	Directory in which to save files [default = tempdir(), unless specified using gl.set.wd].
plot.out	Specify if plot is to be produced [default TRUE].
plot.file	Name for the RDS binary file to save (base name only, exclude extension) [default NULL].
plot_theme	Theme of the plots [default theme_dartR()].
verbose	Verbosity: 0, silent or fatal errors; 1, begin and end; 2, brief progress messages; 3, progress and results summary, including the output of PopCluster; 5, full report [default 2, unless specified using gl.set.verbosity].

Details

For best results, run multiple replicates with different starting seeds to verify convergence and consistency.

Use scaling when your sampling is highly unbalanced (e.g., one population with few individuals vs. another with many). Applying an appropriate scaling level (1, 2, 3, or 4) can substantially improve structure inference in these cases.

If your sample has many closely related individuals, using the Equal Frequency Prior (pr_allele_freq = 1) gives better admixture results. If your sample doesn't include many relatives, the Unequal Frequency Prior (pr_allele_freq = 2) is more accurate. If you're unsure about how related the individuals in your sample are, set pr_allele_freq = 0. This will let the program check for relatedness and automatically choose the best prior (Equal or Unequal) based on the results.

For each K, the Q matrix of the best run chosen by PopCluster is returned; replicates are not averaged.

PopCluster builds carry an expiry date, printed when the program starts (seen with verbose = 3); download a current build when it has passed.

Value

A list with: output_path, the folder with the parameter and input files; best_run, a data frame with one row per K and the numeric columns K, LogL_Mean, LogL_Min, LogL_Max, DLK1, DLK2, FST.FIS (NA where PopCluster reports "-") plus BestRun (the name of the best run); plots, a list of four ggplots (LogL_Mean, DLK1, DLK2, FST.FIS against K); and matrix, a list named by best run of data frames with one row per individual: Index, Order (position in the bar plot), Label, PercentMiss, Cluster, Pop_1 ... Pop_K (ancestry proportions) and Pop (population).

Author(s)

Author(s): Ching Ching Lau. Custodian: Ching Ching Lau – Post to https://groups.google.com/d/forum/dartr

References

- Wang, J. (2022). Fast and accurate population admixture inference from genotype data from a few microsatellites to millions of SNPs. Heredity, 129(2), 79-92.

See Also

Other population structure: gl.evanno(), gl.map.popcluster(), gl.map.snmf(), gl.map.structure(), gl.plot.faststructure(), gl.plot.popcluster(), gl.plot.snmf(), gl.plot.structure(), gl.read.structure(), gl.run.faststructure(), gl.run.snmf()

Examples

```
## Not run:
m <- gl.run.popcluster(x = bandicoot.gl,
  popcluster.path = "/User/PopCluster/Bin/", minK = 1, maxK = 3, rep = 2)
Q <- gl.plot.popcluster(pop_cluster_result = m, plot.K = 3, ind_name = TRUE)
gl.map.popcluster(x = bandicoot.gl, qmat = Q)
# move population 4 (out of 5) 0.5 degrees to the right and population 1
# 0.3 degrees to the south of the map.
mp <- data.frame(lon = c(0, 0, 0, 0.5, 0), lat = c(-0.3, 0, 0, 0, 0))
gl.map.popcluster(bandicoot.gl, qmat = Q, movepops = mp)

## End(Not run)
```

---

gl.run.snmf *Runs a snmf analysis (package LEA) using a genlight object*

Description

Creates an input file for the function snmf (package LEA) and runs it. Refer to the documentation of function snmf for further information on the method and its parameters.

Usage

```
gl.run.snmf(
  x,
  filename = "output",
  minK = 1,
  maxK = 2,
  rep = 1,
  regularization = 10,
  ploidy_lv = 2,
```

```
  ncores = 1,
  cleanup = TRUE,
  plot.out = TRUE,
  plot.dir = NULL,
  plot.file = NULL,
  verbose = NULL,
  ...
)
```

Arguments

x	Name of the genlight object containing the SNP data [required].
filename	File name of output data [default "output"].
minK	Minimum K [default 1].
maxK	Maximum K [default 2].
rep	Number of replicates runs per K [default 1].
regularization	Alpha value for regularization when analyzing small dataset [default 10].
ploidy_lv	Ploidy level of dataset [default 2].
ncores	How many cores should be used [default 1].
cleanup	Remove the LEA run files from the temporary folder when finished [default TRUE].
plot.out	Specify if the cross-entropy plot is to be shown; it is returned either way [default TRUE].
plot.dir	Directory in which to save files [default = tempdir(), unless specified using gl.set.wd].
plot.file	Name for the RDS binary file to save (base name only, exclude extension) [default NULL].
verbose	Verbosity: 0, silent or fatal errors; 1, begin and end; 2, brief progress messages; 3, progress and results summary, including the output of LEA; 5, full report [default 2, unless specified using gl.set.verbosity].
...	Parameters passed to function snmf (package LEA), e.g. seed.

Details

For each K, only the replicate run with the lowest cross-entropy (the best run) is kept; replicates are not averaged.

Value

A list with three elements: best_run, the best run for each K (the path of its folder when cleanup = FALSE, otherwise its name, e.g. "K2/run1"); cross_entropy, a list with the cross-entropy plot (a recordedplot); and matrix, a list named K1, K2, ... of data frames with one row per individual: columns Pop_1 ... Pop_K (ancestry coefficients of the best run), Cluster (the cluster with the largest coefficient), Pop (population), Label (individual name) and Order (position in the bar plot, sorted by population and cluster).

Author(s)

Author(s): Ching Ching Lau. Custodian: Ching Ching Lau – Post to https://groups.google.com/d/forum/dartr

References

- Frichot E, Mathieu F, Trouillon T, Bouchard G, Francois O. (2014). Fast and Efficient Estimation of Individual Ancestry Coefficients. Genetics, 194(4): 973–983.

See Also

Other population structure: gl.evanno(), gl.map.popcluster(), gl.map.snmf(), gl.map.structure(), gl.plot.faststructure(), gl.plot.popcluster(), gl.plot.snmf(), gl.plot.structure(), gl.read.structure(), gl.run.faststructure(), gl.run.popcluster()

Examples

```
## Not run:
m <- gl.run.snmf(x = bandicoot.gl, minK = 1, maxK = 5, rep = 10)
Q <- gl.plot.snmf(snmf.result = m, plot.K = 3, ind.name = TRUE)
gl.map.snmf(bandicoot.gl, qmat = Q)
# move population 4 (out of 5) 0.5 degrees to the right and population 1
# 0.3 degrees to the south of the map.
mp <- data.frame(lon = c(0, 0, 0, 0.5, 0), lat = c(-0.3, 0, 0, 0, 0))
gl.map.snmf(bandicoot.gl, qmat = Q, movepops = mp)

## End(Not run)
```

---

gl.run.stairway2 *Run Stairway Plot 2 for demographic history inference*

Description

This function runs Stairway Plot 2 to infer demographic history using folded SNP frequency spectra. Stairway Plot 2 is a method for inferring demographic history using folded SNP frequency spectra. The key features and methodology of Stairway Plot 2 include:

- **Folded SNP Frequency Spectra**: The method uses folded SNP frequency spectra, which are less sensitive to errors in ancestral state inference compared to unfolded spectra.
- **Demographic Inference**: By analyzing the SNP frequency spectra, Stairway Plot 2 can infer changes in population size over time, providing insights into historical demographic events.
- **Bootstrap Replicates**: The method employs bootstrap replicates to estimate confidence intervals for the inferred demographic history.
- **Flexible Modeling**: Stairway Plot 2 allows for flexible modeling of demographic history without assuming a specific parametric form for population size changes.

To be able to run Stairway Plot 2, the binaries need to be provided in a single folder and can be downloaded via the `gl.download.binary` function. In this case your system needs to have Java installed as well. For more details on the method and how to install on your system refer to the github repository: https://github.com/xiaoming-liu/stairway-plot-v2. Please also refer to the original publication for more details on the method: doi:10.1186/s1305902002196-9. **Also if you use this method, make sure you cite the original publication in your work.** This function implements the theoretical and computational procedures described by Liu and Fu (2020), making it suitable for a wide range of population-genomic datasets to uncover historical demographic patterns.

Usage

```
gl.run.stairway2(
  x,
  L = NULL,
  mu = NULL,
  stairway2.path,
  minbinsize = 1,
  maxbinsize = NULL,
  gentime = 1,
  sfs = NULL,
  parallel = 1,
  run = TRUE,
  blueprint = "blueprint",
  filename = "sample",
  pct_training = 0.67,
  nrand = NULL,
  stairway_plot_dir = "stairway_plot_es",
  nreps = 200,
  seed = NULL,
  plot_title = "Ne",
  xmin = 0,
  xmax = 0,
  ymin = 0,
  ymax = 0,
  xspacing = 2,
  yspacing = 2,
  fontsize = 12,
  cleanup = TRUE,
  plot.display = TRUE,
  plot.theme = theme_dartR(),
  plot.dir = NULL,
  plot.file = NULL,
  verbose = NULL
)
```

Arguments

x	A genlight/dartR object containing SNP data [required].

L	the length of the sequence in base pairs (see details) [default NULL, number of loci x 69; when the sfs is computed from x, only the loci without missing calls, the loci gl.sfs uses, are counted].
mu	the mutation rate per base pair per generation (see details) [required].
stairway2.path	the path to the folder that contains the Stairway Plot 2 folder stairway_plot_es (check the example) [required].
minbinsize	the minimum bin size for the SFS that should be used [default 1].
maxbinsize	the maximum bin size for the SFS that should be used [default NULL, the number of individuals in the dataset].
gentime	the generation time in years [default 1].
sfs	the folded site frequency spectrum (SFS) to be used for the analysis. If not provided the SFS is created from the genlight/dartR object [default NULL].
parallel	the number of parallel processes to use for the analysis [default 1].
run	logical. If TRUE, the analysis is run immediately. Otherwise only the blueprint file is created [might be useful to run on a cluster] [default TRUE].
blueprint	the name of the blueprint file [default "blueprint"].
filename	the name of the population, written as popid in the blueprint and used for the Stairway Plot 2 input file names (no white space) [default "sample"].
pct_training	the proportion of sites to use for training [default 0.67].
nrand	the number of break point settings to try, spread between 0.5 and 2 times (number of individuals - 1). If NULL, the four values recommended by Stairway Plot 2, (nseq-2)/4, (nseq-2)/2, (nseq-2)*3/4 and nseq-2, are used [default NULL].
stairway_plot_dir	the name of the Stairway Plot 2 folder, as written in the blueprint. It must stay "stairway_plot_es", the name of the folder that is copied from stairway2.path [default "stairway_plot_es"].
nreps	the number of bootstrap replicates to use for the analysis [default 200].
seed	the random seed to use for the analysis [default NULL].
plot_title	the title of the plot; also the name of the Stairway Plot 2 output files [default "Ne"].
xmin	minimum x value for the plot [default 0].
xmax	maximum x value for the plot [default 0].
ymin	minimum y value for the plot [default 0].
ymax	maximum y value for the plot [default 0].
xspacing	spacing between x values for the plot [default 2].
yspacing	spacing between y values for the plot [default 2].
fontsize	the font size for the plot [default 12].
cleanup	logical. If TRUE, the folder with the Stairway Plot 2 run files is removed [default TRUE].
plot.display	Specify if plot is to be produced [default TRUE].
plot.theme	User specified theme [default theme_dartR()].

plot.dir	Directory to save the plot RDS files [default as specified by the global working directory or tempdir()]
plot.file	Filename (minus extension) for the RDS plot file [Required for plot save]
verbose	Verbosity: 0, silent or fatal errors; 1, begin and end; 2, brief progress messages; 3, progress and results summary, including the output of Stairway Plot 2; 5, full report [default 2, unless specified using gl.set.verbosity].

Details

Please note: There is currently not really a good way to estimate L, the length of all sequences. Often users of dart data use the number of loci multiplied by 69, but this is definitely an underestimate as monomorphic loci need to be included (also the length of the restriction site should be added for each loci). For mutation rate u, the default value is set to 5e-9, but should be adapted to the species of interest. The good news is, that settings of L and mu affects only the axis of the inferred history, but not the shape of the history. So users can infer the shape, but need to be careful with a temporal interpretation as both x and y axis are affected by the mutation rate and L.

The SFS assumes that every locus is scored in all 2n sequences. A locus with missing calls is counted in a lower frequency class than its true one, so filter loci to full call rate (`gl.filter.callrate(x, threshold = 1)`) or impute them (`gl.impute`) before running the analysis.

Stairway Plot 2 runs in a new subfolder of `tempdir()`. With `cleanup = TRUE` only that subfolder is deleted at the end; with `cleanup = FALSE` or `run = FALSE` it is kept and its path is returned as `run.dir`. With `run = FALSE` the subfolder holds the blueprint file and the Stairway Plot 2 folder, ready to be copied to another machine (e.g. a cluster) and run there with `java -cp stairway_plot_es Stairbuilder blueprint`.

Value

returns a list with three components:

- history: the Stairway Plot 2 summary table: time in mutations per site and in years (`year`), the median effective population size (`Ne_median`) and its 95% (`low95`, `high95`) and 75% (`low75`, `high75`) limits; NULL when `run = FALSE`
- plot: a ggplot of history; NULL when `run = FALSE`
- run.dir: the folder with the Stairway Plot 2 run files; NULL when it was removed (`cleanup = TRUE` and `run = TRUE`)

Author(s)

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

References

Liu, X., Fu, YX. Stairway Plot 2: demographic history inference with folded SNP frequency spectra. Genome Biol 21, 280 (2020). doi:10.1186/s13059020021969

See Also

Other demographic history: `gl.TajimasD()`, `gl.run.epos()`, `gl.sfs()`

Examples

```
## Not run:
#download binary, if not already installed, to tempdir()
gl.download.binary(software="stairway2",os="windows")
require(dartR.data)
sw<- gl.run.stairway2(possums.gl[1:50,1:100], L=1e5, mu = 1e-9,
           stairway2.path = file.path(tempdir(),"stairway2"),
           parallel=5, nreps = 10)
head(sw$history)

## End(Not run)
```

`gl.run.structure` *Runs a STRUCTURE analysis using a genlight object*

Description

This function takes a genlight object and runs a STRUCTURE analysis based on functions from `strataG`

Usage

```
gl.run.structure(
  x,
  exec = "./structure",
  k.range = NULL,
  num.k.rep = 1,
  burnin = 1000,
  numreps = 1000,
  noadmix = TRUE,
  freqscorr = FALSE,
  randomize = TRUE,
  seed = 0,
  pop.prior = NULL,
  locpriorinit = 1,
  maxlocprior = 20,
  gensback = 2,
  migrprior = 0.05,
  pfrompopflagonly = TRUE,
  popflag = NULL,
  inferalpha = FALSE,
  alpha = 1,
  unifprioralpha = TRUE,
  alphamax = 20,
  alphapriora = 0.05,
  alphapriorb = 0.001,
  plot.out = TRUE,
```

```
  plot_theme = theme_dartR(),
  plot.dir = NULL,
  plot.file = NULL,
  delete.files = TRUE,
  verbose = NULL
)
```

Arguments

x	Name of the genlight object containing the SNP data [required].
exec	Full path and name+extension where the structure executable is located. E.g. 'c:/structure/structure.exe' under Windows. For Mac and Linux it might be something like './structure/structure' if the executable is in a subfolder 'structure' in your home directory [default "./structure", i.e. in the working directory].
k.range	Range of the number of populations [default NULL, i.e. from 1 to the number of populations in x].
num.k.rep	Number of replicates [default 1].
burnin	Number of iterations for MCMC burnin [default 1000].
numreps	Number of MCMC replicates [default 1000].
noadmix	Logical. No admixture? [default TRUE].
freqscorr	Logical. Correlated frequencies? [default FALSE].
randomize	Randomize [default TRUE].
seed	Set random seed. Only used when randomize = FALSE; STRUCTURE otherwise seeds from the clock [default 0].
pop.prior	A character specifying which population prior model to use: "locprior" or "usepopinfo" [default NULL].
locpriorinit	Parameterizes locprior parameter r - how informative the populations are. Only used when pop.prior = "locprior" [default 1].
maxlocprior	Specifies range of locprior parameter r. Only used when pop.prior = "locprior" [default 20].
gensback	Integer defining the number of generations back to test for immigrant ancestry. Only used when pop.prior = "usepopinfo" [default 2].
migrprior	Numeric between 0 and 1 listing migration prior. Only used when pop.prior = "usepopinfo" [default 0.05].
pfrompopflagonly	Logical. update allele frequencies from individuals specified by popflag. Only used when pop.prior = "usepopinfo" [default TRUE].
popflag	A vector of integers (0, 1) or logicals identifiying whether or not to use strata information, one per individual in the order of indNames(x). Only used when pop.prior = "usepopinfo" [default NULL].
inferalpha	Logical. Infer the value of the model parameter # from the data; otherwise is fixed at the value alpha which is chosen by the user. This option is ignored under

	the NOADMIX model. Small alpha implies that most individuals are essentially from one population or another, while alpha > 1 implies that most individuals are admixed [default FALSE].
`alpha`	Dirichlet parameter for degree of admixture. This is the initial value if inferalpha = TRUE [default 1].
`unifprioralpha`	Logical. Assume a uniform prior for alpha which runs between 0 and alphamax. This model seems to work fine; the alternative model (when unfprioralpha = 0) is to take alpha as having a Gamma prior, with mean alphapriora x alphapriorb, and variance alphapriora x alphapriorb^2 [default TRUE].
`alphamax`	Maximum for uniform prior on alpha when unifprioralpha = TRUE [default 20].
`alphapriora`	Parameters of Gamma prior on alpha when unifprioralpha = FALSE [default 0.05].
`alphapriorb`	Parameters of Gamma prior on alpha when unifprioralpha = FALSE [default 0.001].
`plot.out`	Create an Evanno plot once finished. The plot needs at least three different values of K in `k.range`; with fewer, the runs are still returned and no plot is drawn [default TRUE].
`plot_theme`	Theme for the plot. See details for options [default theme_dartR()].
`plot.dir`	Directory in which to save the plot RDS file and, when `delete.files = FALSE`, the STRUCTURE files [default tempdir(), or the directory set with gl.set.wd()].
`plot.file`	Name for the RDS binary file to save (base name only, exclude extension) [default NULL].
`delete.files`	logical. Delete all files when STRUCTURE is finished? If FALSE, the input and output files of every run are kept in a time-stamped folder under `plot.dir`; its path is reported at verbose 2 and stored in each run's `files` element [default TRUE].
`verbose`	Verbosity: 0, silent or fatal errors; 1, begin and end; 2, brief progress messages (one line per STRUCTURE run); 3, progress and results summary, including STRUCTURE's own output; 5, full report [default 2, unless specified using gl.set.verbosity].

Details

The function is basically a convenient wrapper around the beautiful strataG function `structureRun` (Archer et al. 2016). For a detailed description please refer to this package (see references below).

Before running STRUCTURE, we suggest reading its manual (see link below) and the literature in mentioned in the references section.

https://web.stanford.edu/group/pritchardlab/structure_software/release_versions/v2.3.4/structure_doc.pdf
To make use of this function you need to download STRUCTURE for you system (**non GUI version**) from here STRUCTURE.

Individual names

STRUCTURE truncates labels at 11 characters and splits them at spaces. Individuals are therefore passed to STRUCTURE by their index and the names in `indNames(x)` are restored in the results, so names of any length, with or without spaces, can be used. Loci that are missing in all individuals are removed before the run.

STRUCTURE runs in a temporary folder; nothing is written to the working directory.

Value

An sr object (structure.result list output). Each list entry is a single structurerun output (there are k.range * num.k.rep number of runs), named k<K>.r<replicate>. For example the summary output of the first run can be accessed via sr[[1]]$summary or the q-matrix of the third run via sr[[3]]$q.mat; rows of q.mat follow the order of indNames(x). To conveniently summarise the outputs across runs (clumpp) you need to run gl.plot.structure on the returned sr object. For Evanno plots run gl.evanno on your sr object.

Author(s)

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

References

- Pritchard, J.K., Stephens, M., Donnelly, P. (2000) Inference of population structure using multilocus genotype data. Genetics 155, 945-959.
- Archer, F. I., Adams, P. E. and Schneiders, B. B. (2016) strataG: An R package for manipulating, summarizing and analysing population genetic data. Mol Ecol Resour. doi:10.1111/1755-0998.12559
- Wang, Jinliang. "The computer program structure for assigning individuals to populations: easy to use but easier to misuse." Molecular ecology resources 17.5 (2017): 981-990.
- Lawson, Daniel J., Lucy Van Dorp, and Daniel Falush. "A tutorial on how not to over-interpret STRUCTURE and ADMIXTURE bar plots." Nature communications 9.1 (2018): 3258.
- Porras-Hurtado, Liliana, et al. "An overview of STRUCTURE: applications, parameter settings, and supporting software." Frontiers in genetics 4 (2013): 98.

Examples

```
# examples need structure to be installed on the system (see above)
## Not run:
bc <- bandicoot.gl[,1:100]
sr <- gl.run.structure(bc, k.range = 2:5, num.k.rep = 3,
exec = './structure.exe')
ev <- gl.evanno(sr)
ev
qmat <- gl.plot.structure(sr, K=3)
head(qmat)
gl.map.structure(qmat, bc, scalex=1, scaley=0.5)

## End(Not run)
```

`gl.select.panel` *Select a panel of loci based on various methods*

Description

This function selects a panel of loci from a SNP genlight object based on various selection methods, for example to design a targeted genotyping panel. Use `gl.check.panel` to compare the panel with the full data.

Usage

```
gl.select.panel(
  x,
  method = "random",
  nl = 10,
  exact = TRUE,
  plot.out = TRUE,
  plot.file = NULL,
  plot.dir = NULL,
  verbose = NULL,
  include.loci = NULL
)
```

Arguments

`x`	A genlight object with SNP data [required].
`method`	The selection method, one of "random", "dapc", "pahigh", "monopop", "stratified", "hafall", "hafpop", "pic" or "picdart" (see Details) [default "random"].
`nl`	The number of loci to select, between 1 and nLoc(x) [default 10].
`exact`	Logical. If TRUE, the panel has exactly `nl` loci: surplus loci are dropped at random, and missing ones are added at random from the remaining loci. If FALSE, the panel holds what the method selected, which can be more or fewer than `nl`; the function stops if the method selected no loci [default TRUE].
`plot.out`	Not used: the function draws no plot. Accepted so that existing calls keep working [default TRUE].
`plot.file`	Not used, see `plot.out` [default NULL].
`plot.dir`	Not used, see `plot.out` [default NULL].
`verbose`	Verbosity: 0, silent or fatal errors; 1, begin and end; 2, brief progress messages; 3, progress and results summary; 5, full report [default 2, unless specified using gl.set.verbosity].
`include.loci`	Names of loci to force into the panel whatever the method, for example from a legacy panel. They are always kept, and the method fills the remaining `nl - length(include.loci)` places. It must not hold more than `nl` loci [default NULL].

Details

Each method has specific criteria for selecting loci:

- "random": selects loci at random.
- "dapc": for each pair of populations, runs a DAPC and takes the ceiling(nl / number of pairs) loci contributing most to the first discriminant function.
- "pahigh": for each pair of populations and each direction, takes the ceiling(nl / (2 * number of pairs)) private-allele loci whose private alleles have the highest frequency.
- "monopop": for each population, takes up to ceiling(nl / number of populations) loci at random among those monomorphic in that population.
- "stratified": for each population, bins loci by minor allele frequency into ceiling(nl / number of populations) classes between 0 and 0.5 and takes one locus at random from each class.
- "hafall": takes the loci with the highest minor allele frequency (closest to 0.5) across all individuals.
- "hafpop": for each population, takes the ceiling(nl / number of populations) loci with the highest minor allele frequency in that population.
- "pic": takes the loci with the highest polymorphic information content, 1 - (p^2 + q^2) - 2 p^2 q^2.
- "picdart": takes the loci with the highest AvgPIC, recalculated with gl.recalc.metrics.

The per-population and per-pair methods can select more than `nl` loci (rounding up, then combining); with `exact = TRUE` the surplus is dropped at random, so the ranking is not used to choose which loci go. "random", "monopop", "stratified" and the `exact` adjustment use R's random number generator; call `set.seed()` first for a reproducible panel.

All methods except "random", "hafall", "pic" and "picdart" need populations; "dapc" and "pahigh" need at least two. "dapc" leaves out, for each pair, the loci with no calls in that pair.

Value

A genlight object with the selected loci; individuals are in the same order as in `x`.

Author(s)

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

Examples

```
# Select 50 loci randomly
selected <- gl.select.panel(possums.gl, method = "random", nl = 50)

# Select 5 loci based on DAPC
selected <- gl.select.panel(possums.gl, method = "dapc", nl = 5)
```

`gl.select.panel.combined` *Select a SNP panel optimised for specified conservation monitoring parameters*

Description

Selects a panel of `nl` loci optimised simultaneously across a user-specified set of conservation monitoring parameters using simulated annealing (SA) with adaptive swapping and optional restart-from-best. Press Escape to stop — the best panel found is returned cleanly.

Usage

```
gl.select.panel.combined(
  x,
  nl = 10,
  parameter = c("Fst", "He"),
  neest.path = NULL,
  corr.method = c("spearman", "pearson"),
  ref = c("global", "by.pop"),
  metric = c("grm", "kinship"),
  inverse_dr = FALSE,
  sample = 1,
  include.loci = NULL,
  weights = NULL,
  init.method = "random",
  n_iter = 100,
  start_temp = 0.1,
  cooling = NULL,
  restart_tol = NULL,
  stop.criterion = NULL,
  error.rate = 0.01,
  threshold = 0.001,
  n_sim_parents = 50,
  n_sim_hyb = 100,
  plot.out = TRUE,
  plot.every = 10,
  plot.file = NULL,
  plot.dir = NULL,
  verbose = NULL
)
```

Arguments

`x`	A `dartR` or `genlight` object. Full dataset from which the panel is selected and against which performance is evaluated. If `"hybridisation"` is in `parameter`, `x` must have at least 2 populations; all pairwise population crosses are evaluated.

`nl` Integer. Number of loci to select. Default `10`.

`parameter` Character vector. Parameters to optimise. Subset of `"Fst"`, `"He"`, `"Ho"`, `"Fis"`, `"Nall"`, `"Ho_ind"`, `"relatedness"`, `"id"`, `"parentage"`, `"assignment"`, `"hybridisation"`, `"Ne"`. The shortcuts `"all"` (all parameters except `"hybridisation"`) and `"all-Ne"` (as `"all"` but also excluding `"Ne"`) are expanded automatically. Default `c("Fst", "He")`.

`neest.path` Character string. Path to NEstimator executable, required only when `"Ne"` is included in `parameter`.

`corr.method` Character. Correlation method passed to `gl.check.panel` for correlation-based performance metrics. Options are `"spearman"` (default) and `"pearson"`.

`ref` Character. Allele-frequency reference for `"relatedness"`, passed to `gl.check.panel`. `"global"` (default) pools across the dataset and uses all pairs; `"by.pop"` uses within-population frequencies and within-population pairs only. Affects the relatedness objective when `"relatedness"` is optimised.

`metric` Character. Relatedness reporting scale passed to `gl.check.panel`: `"grm"` (default) or `"kinship"` (GRM/2). As a constant rescaling it does not change the optimisation objective; recorded for reproducibility.

`inverse_dr` Logical. If `TRUE` and `"drift_resistance"` is among `parameter`, the performance is reported as `1 - drift_resistance` (drift sensitivity). The SA then optimises for loci with *low* MAF, producing a panel that is maximally sensitive to allele-frequency change. Default `FALSE`.

`sample` Numeric in (0,1]. Proportion of individuals per population to retain from the panel when evaluating performance during optimisation (passed to `gl.check.panel`). `1` (default) uses all individuals; the reference dataset is always full size.

`include.loci` Character vector of locus names to force into the panel and keep fixed throughout optimisation (e.g. loci from a legacy panel). These are never swapped out by the SA; only the remaining `nl - length(include.loci)` slots are optimised. Must not exceed `nl`. Default `NULL`.

`weights` Numeric vector, one per element of `parameter`. `NULL` (default) = equal weights.

`init.method` Initialisation method from `gl.select.panel`. Default `"random"`.

`n_iter` Integer. Maximum SA iterations. Default `100`.

`start_temp` Numeric. Starting temperature. Default `0.1`.

`cooling` Numeric (0,1) or `NULL`. Geometric cooling rate per iteration. If `NULL` (default), computed automatically:

$$\text{cooling} = \left(\frac{0.001}{\text{start_temp}}\right)^{1/\text{n_iter}}$$

Set explicitly to override.

`restart_tol` Numeric or `NULL`. Restart-from-best tolerance. If the current composite performance falls more than `restart_tol` below the best performance seen so far, the current panel is reset to the best panel before the next iteration. This prevents the search from wasting iterations in low-performance regions. `NULL` (default) disables restarting. A value of `0.1` is a reasonable starting point.

stop.criterion	Numeric (0,1) or NULL. Auto-stop when composite performance reaches this value. Default NULL.
error.rate	Numeric. Per-allele error rate for "id", "parentage", "assignment", "hybridisation". Default 0.01.
threshold	Numeric. Confusion threshold for "id" and "parentage". Default 0.001.
n_sim_parents	Integer. Number of simulated trios for "parentage". Default 50.
n_sim_hyb	Integer. Simulated F1 individuals per pairwise population cross for "hybridisation". Default 100.
plot.out	Logical. Live convergence plot refreshed every plot.every iterations. Default TRUE.
plot.every	Integer. Plot refresh interval in iterations. Default 10.
plot.file	Character string. File name for saving the final plot.
plot.dir	Character string. Directory for saving the final plot.
verbose	Integer controlling verbosity.

Details

Adaptive swapping: The number of loci swapped per iteration scales linearly with temperature:

$$n_swap(t) = \max(1,\ \text{round}(\lfloor 0.3 \cdot nl \rceil \cdot T(t)/\text{start_temp}))$$

Up to 30% of the panel is replaced early (broad exploration); by the end only 1 locus is swapped (fine-tuning). Following Ingber (1989).

Restart from best: When restart_tol is set, after each accept/reject step, if curr_perf < best_perf - restart_tol the current panel is reset to the best panel found so far. This anchors exploration near the best solution without disabling the temperature-controlled acceptance of new proposals. Restarts are counted and reported.

Auto-cooling: When cooling = NULL the rate is chosen automatically to match n_iter.

Composite performance:

$$\text{composite} = \frac{\sum_p w_p \cdot \text{performance}_p}{\sum_p w_p}$$

Value

A dartR or genlight object with nl selected loci. Stored in x@other:

- sa_best_performance
- sa_history — data frame: iteration, current_performance, best_performance, drift_resistance (per-iteration drift-resistance score, NA if not in the parameter set), other_performance (weighted mean of all non-drift-resistance parameters, renormalised to sum to 1), n_swap, restarted
- sa_parameters — list: parameter, weights, neest.path, corr.method, ref, metric, cooling, n_swap_max, restart_tol
- sa_stop_reason — "completed", "stop_criterion", or "interrupted"

References

Ingber, L. (1989). Very fast simulated re-annealing. *Mathematical and Computer Modelling*, 12(8), 967–973.

Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, 220(4598), 671–680.

Examples

```
# Short demonstration on four populations; use n_iter of 100 or more in real analyses
x <- gl.keep.pop(possums.gl, pop.list = levels(pop(possums.gl))[1:4], verbose = 0)
panel <- gl.select.panel.combined(x, nl = 20,
                                  parameter = c("Fst", "He"),
                                  n_iter = 3, plot.every = 3)

# With restart-from-best
panel <- gl.select.panel.combined(x, nl = 20,
                                  parameter    = c("Fst", "He", "Fis"),
                                  weights      = c(2, 1, 1),
                                  n_iter       = 3,
                                  plot.every   = 3,
                                  restart_tol  = 0.1)

panel@other$sa_best_performance
panel@other$sa_stop_reason
```

---

gl.select.panel.parallel

Run multiple simulated-annealing chains in parallel with synchronized live plotting (burst-sync), returning the single best panel found.

Description

Runs `n_chains` independent SA optimisations of the same objective as `gl.select.panel.combined`, each on its own core. Chains advance in synchronized bursts: every `burst` iterations all chains pause, the master collects their progress, redraws a combined convergence plot showing every chain, then signals all chains to resume. This lets you watch whether the chains are converging to a common performance level (useful when you want confidence the panel has plateaued).

Usage

```
gl.select.panel.parallel(
  x,
  nl = 10,
  parameter = c("Fst", "He"),
  n_chains = 4,
  n_cores = NULL,
```

```
  n_iter = 200,
  burst = 10,
  neest.path = NULL,
  corr.method = c("spearman", "pearson"),
  ref = c("global", "by.pop"),
  metric = c("grm", "kinship"),
  inverse_dr = FALSE,
  sample = 1,
  include.loci = NULL,
  weights = NULL,
  init.method = "random",
  start_temp = 0.1,
  cooling = NULL,
  restart_tol = NULL,
  stop.criterion = NULL,
  error.rate = 0.01,
  threshold = 0.001,
  n_sim_parents = 50,
  n_sim_hyb = 100,
  plot.out = TRUE,
  verbose = 1
)
```

Arguments

x	A genlight object.
nl	Panel size (number of loci).
parameter	Character vector of parameters to optimise (see gl.select.panel.combined).
n_chains	Number of parallel SA chains (recommended 3-5).
n_cores	Cores to use. Default = n_chains.
n_iter	Total SA iterations per chain.
burst	Iterations each chain runs between synchronization/plot points. Smaller = smoother live plot but more overhead. Default 10.
corr.method, ref, metric, inverse_dr, sample, include.loci, error.rate, threshold, n_sim_parents, n_sim_hyb, neest.path	As in gl.select.panel.combined, passed to each chain's evaluator.
weights, init.method, start_temp, cooling, restart_tol, stop.criterion	As in gl.select.panel.combined. Each chain is seeded from a different random panel (init.method applied per chain with a distinct seed) unless include.loci forces some loci.
plot.out	Logical. Draw the combined live plot at each burst.
verbose	0/1/2.

Details

Interruptible: pressing Escape stops all workers cleanly and returns the best panel found across all chains up to the last completed burst.

Value

A genlight (the single best panel across all chains) with the same `@other$sa_*` metadata slots as `gl.select.panel.combined`, plus `@other$sa_all_chains` (per-chain best performance and history).

`gl.sfs` *Creates a site frequency spectrum based on a dartR or genlight object*

Description

Creates a folded or unfolded site frequency spectrum (SFS), for all individuals together or as a multidimensional (joint) SFS with one dimension per population.

Usage

```
gl.sfs(
  x,
  minbinsize = 0,
  folded = TRUE,
  singlepop = FALSE,
  plot.out = TRUE,
  plot.file = NULL,
  plot.dir = NULL,
  plot.theme = theme_dartR(),
  verbose = NULL
)
```

Arguments

`x`	dartR/genlight object with SNP data [required].
`minbinsize`	remove bins from the left of the sfs. For example to remove singletons (alleles only occurring once among all individuals) set minbinsize to 2. If set to zero, also monomorphic (d0) loci are returned. For a multidimensional sfs see details [default 0].
`folded`	if set to TRUE (default) a folded sfs (minor allele frequency sfs) is returned. If set to FALSE then an unfolded (derived allele frequency sfs) is returned. It is assumed that 0 is homozygote for the reference and 2 is homozygote for the derived allele. So you need to make sure your coding is correct [default TRUE].
`singlepop`	switch to force to create a one-dimensional sfs, even though the genlight/dartR object contains more than one population [default FALSE].
`plot.out`	Specify if plot is to be produced [default TRUE].
`plot.file`	Name for the RDS binary file to save (base name only, exclude extension); only a one-dimensional sfs is plotted [default NULL]
`plot.dir`	Directory in which to save files [default as specified by the global working directory or tempdir()]

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

Details

Only loci scored in every individual are used. A locus with missing calls cannot be placed in a frequency class of the full sample: counted against the full sample size it lands in a lower class, and a locus fixed in all called individuals would appear polymorphic. Loci with missing calls are therefore excluded and their number is reported (verbose >= 1); to keep more loci, filter individuals with low call rate or impute (gl.impute) before running the function.
A folded multidimensional SFS is folded on the minor allele of all populations combined (the joint minor allele frequency SFS of fastsimcoal2): for each locus, if the alternative allele is the more frequent one over all individuals, the counts of every population are taken for the reference allele.
For a multidimensional SFS, minbinsize sets to zero the cells whose total count over all populations is below minbinsize (for minbinsize = 1, only the cell of loci monomorphic in all populations); the array keeps its full dimensions, so polymorphisms private to one population are kept.

Value

returns a site frequency spectrum, either a one dimensional vector (only a single population in the dartR/genlight object or singlepop=TRUE) or an n-dimensional array (n is the number of populations in the genlight/dartR object). If the dartR/genlight object consists of several populations the multidimensional site frequency spectrum for each population is returned [=a multidimensional site frequency spectrum]. Be aware the multidimensional spectrum works only for a limited number of population and individuals [if too high the function stops with an error as the number of populations and individuals (and therefore dimensions) are too large]. To get a single sfs for a genlight/dartR object with multiple populations, you need to set singlepop to TRUE. The returned sfs can be used to analyse demographics, e.g. using fastsimcoal2.

Author(s)

Author(s): Bernd Gruber. Custodian: Bernd Gruber & Carlo Pacioni – Post to https://groups.google.com/d/forum/dartr

References

Excoffier L., Dupanloup I., Huerta-Sanchez E., Sousa V. C. and Foll M. (2013) Robust demographic inference from genomic and SNP data. PLoS genetics 9(10)

See Also

Other demographic history: gl.TajimasD(), gl.run.epos(), gl.run.stairway2()

Examples

```
gl.sfs(bandicoot.gl, singlepop = TRUE)
gl.sfs(possums.gl[c(1:5, 31:33), ], minbinsize = 1)
```

gl.TajimasD *Calculation of Tajima's D*

Description

This function calculates Tajima's D for each population in the dartR/genlight object, with p-values from the beta distribution, the standard normal distribution and, if rep is given, from simulations of a neutral population with unlinked SNPs.

Usage

```
gl.TajimasD(
  x,
  ms.path = NULL,
  simulation.out = NULL,
  rep = NULL,
  seeds = NULL,
  cleanup = TRUE,
  plot.dir = NULL,
  plot.out = TRUE,
  plot.file = NULL,
  plot_theme = NULL,
  verbose = NULL
)
```

Arguments

x	Name of the genlight object containing the SNP data [required].
ms.path	Not needed any more: the neutral simulation is done in R. Accepted for compatibility and ignored [default NULL].
simulation.out	Directory in which to save the simulated Tajima's D values (one file per population), if rep is given [default NULL].
rep	Number of simulated replicates for sim_pval. If NULL, no simulation is run [default NULL].
seeds	Seed for the random number generator of the simulation (the first value is used) [default NULL, not set].
cleanup	Not needed any more (no temporary files are written); accepted for compatibility [default TRUE].
plot.dir	Directory in which to save files [default as specified by the global working directory or tempdir()].
plot.out	Specify if plot is to be produced [default TRUE].
plot.file	Name for the RDS binary file to save (base name only, exclude extension) [default NULL].
plot_theme	Theme of the plot [default theme_dartR()].
verbose	Verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log; 3, progress and results summary; 5, full report [default 2 or as specified using gl.set.verbosity].

Details

D is calculated from the allele counts at each locus: pi (the average number of pairwise differences) and Watterson's estimator use the number of called sequences at each locus, so loci with missing calls are used with their own sample size. The constants in the variance of D use the mean sample size (reported as N, the mean number of genotyped individuals).
Pval.normal and Pval.beta follow Tajima (1989), which assumes that all segregating sites lie on one non-recombining locus. SNPs from DArT data are mostly unlinked, and for unlinked sites D varies much less under neutrality, so these two p-values are too large (conservative) for such data. sim_pval is computed against a neutral null of unlinked SNPs: for each replicate, the derived-allele count of each of the S segregating sites is drawn from the neutral distribution P(k) proportional to 1/k (k = 1, ..., 2N - 1), and D is computed from the simulated sites. sim_pval is the proportion of replicates with |D| at least as large as the observed |D|.
DArT loci are selected because they are polymorphic, which shifts the site frequency spectrum towards intermediate frequencies and D upwards; interpret D with this ascertainment in mind.

Value

A data frame with, for each population, pi, the number of segregating sites S, D, Pval.normal, Pval.beta, N (mean number of genotyped individuals), theta_per_site (pi divided by the number of loci, not by the number of base pairs) and, if rep is given, sim_pval. A plot of D by population and, if rep is given, the simulated distributions are shown.

Author(s)

Author(s): Renee Catullo. Custodian: Ching Ching Lau – Post to https://groups.google.com/d/forum/dartr

References

- Tajima, F. (1989). Statistical method for testing the neutral mutation hypothesis by DNA polymorphism. Genetics, 123(3), 585-595.
- Fu, Y. X. (1995). Statistical properties of segregating sites. Theoretical Population Biology, 48(2), 172-197.
- Paradis, E. (2010). pegas: an R package for population genetics with an integrated-modular approach. Bioinformatics, 26(3), 419-420.

See Also

Other demographic history: gl.run.epos(), gl.run.stairway2(), gl.sfs()

Examples

```
Tajima <- gl.TajimasD(x=bandicoot.gl)
# with a neutral null of unlinked SNPs

Tajima <- gl.TajimasD(x=bandicoot.gl, rep=100, seeds=1)
```

`utils.get.allele.freq` *utils.get.allele.freq*

Description

Generates percentage allele frequencies by locus and population – This is copy from package dartR function gl.percent.freq

Usage

```
utils.get.allele.freq(x, verbose = NULL)
```

Arguments

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

Details

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

The frequency column is the percentage of the alternative allele (or of presence), rounded to 2 decimals; use sum and nobs for exact values.

Value

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

Author(s)

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

Examples

```
m <-  utils.get.allele.freq(testset.gl)
```

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

Description

This function is the original implementation of Outflank by Whitlock and Lotterhos. dartR simply provides a convenient wrapper around their functions and an easier install being an r package (for information please refer to their github repository)

Usage

```
utils.outflank(
  FstDataFrame,
  LeftTrimFraction = 0.05,
  RightTrimFraction = 0.05,
  Hmin = 0.1,
  NumberOfSamples,
  qthreshold = 0.05,
  verbose = NULL
)
```

Arguments

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

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

`LeftTrimFraction`
The proportion of loci that are trimmed from the lower end of the range of Fst before the likelihood funciton is applied [default 0.05].

`RightTrimFraction`
The proportion of loci that are trimmed from the upper end of the range of Fst before the likelihood funciton is applied [default 0.05].

Hmin	The minimum heterozygosity required before including calculations from a locus [default 0.1].
NumberOfSamples	The number of spatial locations included in the data set.
qthreshold	The desired false discovery rate threshold for calculating q-values [default 0.05].
verbose	Verbosity: 0, silent or fatal errors; 1, begin and end; 2, brief progress messages; 3, progress and results summary; 5, full report [default 2, unless specified using gl.set.verbosity].

Details

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:

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

---

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

Description

Creates OutFLANK input file from individual genotype info.

Usage

```
utils.outflank.MakeDiploidFSTMat(SNPmat, locusNames, popNames, verbose = NULL)
```

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

Value

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

`utils.outflank.plotter` *Plotting functions for Fst distributions after OutFLANK*

Description

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

Usage

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

Arguments

`OFoutput`	The output of the function OutFLANK()
`withOutliers`	Determines whether the loci marked as outliers (with $OutlierFlag) are included in the histogram.
`NoCorr`	Plots the distribution of FSTNoCorr when TRUE. Recommended, because this is the data used by OutFLANK to infer the distribution.
`Hmin`	The minimum heterozygosity required before including a locus in the plot.
`binwidth`	The width of bins in the histogram.
`Zoom`	If Zoom is set to TRUE, then the graph will zoom in on the right tail of the distirbution (based on argument RightZoomFraction)
`RightZoomFraction`	Used when Zoom = TRUE. Defines the proportion of the distribution to plot.
`titletext`	Allows a test string to be printed as a title on the graph

Value

produces a histogram of the FST

utils.structure.evanno

Evanno statistics and plots from a STRUCTURE run object

Description

Computes the Evanno et al. (2005) statistics from a structure run object (gl.run.structure) and builds their plots. Called by gl.evanno and gl.run.structure.

The code was adapted from package strataG (Eric Archer), which is no longer on CRAN. See gl.evanno for the statistics returned.

Usage

```
utils.structure.evanno(sr, plot = TRUE, plot.theme = NULL, verbose = 0)
```

Arguments

sr	Structure run object from gl.run.structure [required].
plot	Whether the combined plot is printed [default TRUE].
plot.theme	Theme added to every plot; NULL leaves the ggplot2 default [default NULL].
verbose	Verbosity: 0, silent or fatal errors; 1, begin and end; 2, brief progress messages; 3, progress and results summary; 5, full report [default 0].

Value

A list with the statistics (element df) and a list of plots (element plots: mean.ln.k, ln.pk, ln.ppk, delta.k when it can be computed, and combined).

Author(s)

Author(s): Bernd Gruber, original implementation by Eric Archer (https://github.com/EricArcher/strataG). Custodian: Bernd Gruber – Post to https://groups.google.com/d/forum/dartr

utils.structure.genind2gtypes

structure util functions

Description

These functions were copied from package strataG, which is no longer on CRAN (maintained by Eric Archer)

Usage

```
utils.structure.genind2gtypes(x)
```

Arguments

x	a genind object

Value

a gtypes object

Author(s)

Bernd Gruber (bugs? Post to https://groups.google.com/d/forum/dartr); original implementation of Eric Archer https://github.com/EricArcher/strataG

---

utils.structure.run *Utility function to run Structure*

Description

These functions were copied from package strataG, which is no longer on CRAN (maintained by Eric Archer)

Usage

```
utils.structure.run(
  g,
  k.range,
  num.k.rep,
  label,
  delete.files = TRUE,
  exec,
  burnin,
  numreps,
  noadmix,
  freqscorr,
  randomize,
  seed,
  pop.prior,
  locpriorinit,
  maxlocprior,
  gensback,
  migrprior,
  pfrompopflagonly,
  popflag,
  inferalpha,
  alpha,
  unifprioralpha,
  alphamax,
  alphapriora,
```

```
  alphapriorb,
  ind.names = NULL,
  keep.dir = NULL,
  verbose = 0
)
```

Arguments

g	a gtypes object [see strataG].
k.range	vector of values to for maxpop in multiple runs. If set to NULL, a single STRUCTURE run is conducted with maxpops groups. If specified, do not also specify maxpops.
num.k.rep	number of replicates for each value in k.range.
label	label to use for input and output files (currently unused; runs are labelled k<K>.r<replicate>).
delete.files	logical. Delete all files when STRUCTURE is finished? If FALSE, the files are kept in a time-stamped folder under keep.dir.
exec	name of executable for STRUCTURE. Defaults to "structure".
burnin	Number of burnin reps [default 10000].
numreps	Number of MCMC replicates [default 1000].
noadmix	Logical. No admixture? [default TRUE].
freqscorr	Logical. Correlated frequencies? [default FALSE].
randomize	Randomize [default TRUE].
seed	Set random seed [default 0].
pop.prior	A character specifying which population prior model to use: "locprior" or "usepopinfo" [default NULL].
locpriorinit	Parameterizes locprior parameter r - how informative the populations are. Only used when pop.prior = "locprior" [default 1].
maxlocprior	Specifies range of locprior parameter r. Only used when pop.prior = "locprior" [default 20].
gensback	Integer defining the number of generations back to test for immigrant ancestry. Only used when pop.prior = "usepopinfo" [default 2].
migrprior	Numeric between 0 and 1 listing migration prior. Only used when pop.prior = "usepopinfo" [default 0.05].
pfrompopflagonly	Logical. update allele frequencies from individuals specified by popflag. Only used when pop.prior = "usepopinfo" [default TRUE].
popflag	A vector of integers (0, 1) or logicals identifiying whether or not to use strata information, one per individual in the order of ind.names (or of the sorted ids when ind.names is NULL). Only used when pop.prior = "usepopinfo" [default NULL].
inferalpha	Logical. Infer the value of the model parameter # from the data; otherwise is fixed at the value alpha which is chosen by the user. This option is ignored under the NOADMIX model. Small alpha implies that most individuals are essentially from one population or another, while alpha > 1 implies that most individuals are admixed [default FALSE].

`alpha` Dirichlet parameter for degree of admixture. This is the initial value if inferalpha = TRUE [default 1].

`unifprioralpha` Logical. Assume a uniform prior for alpha which runs between 0 and alphamax. This model seems to work fine; the alternative model (when unfprioralpha = 0) is to take alpha as having a Gamma prior, with mean alphapriora x alphapriorb, and variance alphapriora x alphapriorb^2 [default TRUE].

`alphamax` Maximum for uniform prior on alpha when unifprioralpha = TRUE [default 20].

`alphapriora` Parameters of Gamma prior on alpha when unifprioralpha = FALSE [default 0.05].

`alphapriorb` Parameters of Gamma prior on alpha when unifprioralpha = FALSE [default 0.001].

`ind.names` Individual names in the order they should appear in the results. When given, individuals are passed to STRUCTURE by their index (STRUCTURE truncates labels at 11 characters and rejects spaces) and the names are restored in `q.mat` and `prior.anc`; `q.mat` is returned in this order [default NULL, ids passed as they are].

`keep.dir` Directory under which the STRUCTURE files are kept when `delete.files = FALSE` [default tempdir()].

`verbose` Verbosity: 0, silent; 2, one line per run; 3 and above, STRUCTURE's own output is shown [default 0].

Value

`structureRun` a list where each element is a list with results from `utils.structure.read` and a vector of the filenames used

`structureWrite` a vector of the filenames used by STRUCTURE

`utils.structure.read` a list containing:

- `summary` new locus name, which is a combination of loci in group
- `q.mat` data.frame of assignment probabilities for each id
- `prior.anc` list of prior ancestry estimates for each individual where population priors were used
- `files` vector of input and output files used by STRUCTURE
- `label` label for the run

Author(s)

Bernd Gruber (bugs? Post to https://groups.google.com/d/forum/dartr); original implementation of Eric Archer https://github.com/EricArcher/strataG

Index

∗ **annotation and mapping helpers**
gl.find.genes.for.loci, 26
gl.find.loci.in.genes, 28
∗ **assignment**
gl.assign.mahalanobis, 4
gl.assign.on.genotype, 7
gl.assign.pa, 9
gl.assign.pca, 10
∗ **basic statistics**
gl.amova, 3
∗ **demographic history**
gl.run.epos, 69
gl.run.stairway2, 82
gl.sfs, 97
gl.TajimasD, 99
∗ **fixed difference analysis**
gl.collapse, 23
∗ **hybridisation**
gl.run.mongrail, 75
∗ **ld functions**
gl.ld.distance, 30
gl.ld.haplotype, 32
∗ **linker**
gl.run.assignpop, 63
∗ **panel selection**
gl.select.panel, 90
∗ **population genetics**
gl.opt.ind.all, 46
∗ **population structure**
gl.evanno, 24
gl.map.popcluster, 37
gl.map.snmf, 39
gl.map.structure, 41
gl.plot.faststructure, 51
gl.plot.popcluster, 54
gl.plot.snmf, 56
gl.plot.structure, 58
gl.read.structure, 61
gl.run.faststructure, 73
gl.run.popcluster, 77
gl.run.snmf, 80
∗ **reference genomes**
gl.blast, 12
∗ **selection**
gl.outflank, 49
gl.run.bayescan, 65

gl.amova, 3
gl.assign.mahalanobis, 4, *8*, *10*, *12*
gl.assign.on.genotype, *6*, 7, *10*, *12*
gl.assign.pa, *6*, *8*, 9, *12*
gl.assign.pca, *6*, *8*, *10*, 10
gl.blast, 12
gl.check.future.panel, 16
gl.check.panel, *17*, 19, *90*
gl.collapse, 23
gl.evanno, 24, *39*, *41*, *43*, *54, 55*, *58*, *60–62*, *75*, *80*, *82*, *106*
gl.filter.callrate, *49*
gl.filter.locmetric, *15*
gl.find.genes.for.loci, 26, *29*
gl.find.loci.in.genes, *28*, 28
gl.ld.distance, 30, *34*
gl.ld.haplotype, *31*, 32
gl.LDNe, 35
gl.map.popcluster, *26*, 37, *41*, *43*, *54, 55*, *58*, *60*, *62*, *75*, *80*, *82*
gl.map.snmf, *26*, *38, 39*, 39, *43*, *54, 55*, *58*, *60*, *62*, *75*, *80*, *82*
gl.map.structure, *26*, *38–41*, 41, *54, 55*, *58*, *60–62*, *75*, *80*, *82*
gl.nhybrids, 44
gl.opt.ind.all, 46
gl.outflank, 49, *68*
gl.plot.faststructure, *26*, *39*, *41*, *43*, 51, *55*, *58*, *60*, *62*, *75*, *80*, *82*
gl.plot.popcluster, *26*, *37–39*, *41*, *43*, *54*, 54, *58*, *60*, *62*, *75*, *80*, *82*

gl.plot.snmf, *26*, *39–41*, *43*, *54, 55*, 56, *60*, *62*, *75*, *80*, *82*
gl.plot.structure, *26*, *39*, *41–43*, *51*, *53–55*, *58*, 58, *61, 62*, *75*, *80*, *82*
gl.read.structure, *26*, *39*, *41*, *43*, *54, 55*, *58*, *60*, 61, *75*, *80*, *82*
gl.report.ld.map, *30, 31*
gl.run.assignpop, 63
gl.run.bayescan, *51*, 65
gl.run.epos, 69, *85*, *98*, *100*
gl.run.faststructure, *26*, *39*, *41*, *43*, *51, 52*, *54, 55*, *58*, *60*, *62*, 73, *80*, *82*
gl.run.mongrail, 75
gl.run.popcluster, *26*, *37*, *39*, *41*, *43*, *54, 55*, *58*, *60*, *62*, *75*, 77, *82*
gl.run.snmf, *26*, *39–41*, *43*, *54–56*, *58*, *60*, *62*, *75*, *80*, 80
gl.run.stairway2, *72*, 82, *98*, *100*
gl.run.structure, *24–26*, *41*, *43*, *58–62*, 86, *106*
gl.select.panel, 90, *93*
gl.select.panel.combined, 92, *95*
gl.select.panel.parallel, 95
gl.sfs, *72*, *85*, 97, *100*
gl.TajimasD, *72*, *85*, *98*, 99
gl2bayescan, *68*

ld, *33*

providers, *38*, *40*, *42*

scale_fill_viridis, *34*
snmf, *80, 81*

utils.get.allele.freq, 101
utils.outflank, *51*, 102
utils.outflank.MakeDiploidFSTMat, *51*, 104
utils.outflank.plotter, *51*, 105
utils.structure.evanno, 106
utils.structure.genind2gtypes, 106
utils.structure.run, 107