Package: dartR.captive (via r-universe)

November 20, 2026

Type Package

Title Analysing 'SNP' Data to Support Captive Breeding

Version 1.3.1

Revision Hardy Hedgehog

Date 2026-11-18

Description Functions are provided that facilitate the analysis of SNP (single nucleotide polymorphism) data to answer questions regarding captive breeding and relatedness between individuals. 'dartR.captive' is part of 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 (>= 3.5), dartR.base (>= 1.3.0), dartR.data (>= 1.2.5), dartR.sim

Imports adegenet (>= 2.0.0), methods, utils, crayon, ggplot2, patchwork, stringr, data.table, gridExtra, magrittr,reshape2,tidyr, processx (>= 3.3.0)

Suggests poibin, dartR.popgen, SIBER, gplots, fields, igraph, kinship2, rrBLUP, scales, spelling, testthat (>= 3.0.0), tidyverse, 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], Peter J. Unmack [ctb], Oliver Berry [ctb], Lindsay V. Clark [ctb], Floriaan Devloo-Delva [ctb], Eric Archer [ctb], Sam Amini [ctb], Ethan Halford [ctb], Jinliang Wang [ctb] (author of Coancestry, on which gl.relatedness is based)

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

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

Language en-US

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

Config/pak/sysreqs zlib1g-dev

Repository https://cran.r-universe.dev

Date/Publication 2026-11-19 23:00:02 UTC

RemoteUrl https://github.com/cran/dartR.captive

RemoteRef HEAD

RemoteSha 15d411d00abd5c5985f53c24162f3edc9678939b

Contents

gl.assign.grm 2
gl.diagnostics.relatedness 3
gl.filter.parent.offspring 9
gl.grm 11
gl.grm.network 13
gl.inbreeding 17
gl.parentage.power 20
gl.plot.network 23
gl.relatedness 25
gl.report.errorrate 28
gl.report.parent.offspring 30
gl.run.colony 33
gl.run.EMIBD9 37
gl.sim.relatedness 40
gl.test.offspring 43
gl2colony 46

Index **50**

`gl.assign.grm` *Population assignment using grm*

Description

This function takes one individual and, for each candidate population, estimates their mean relatedness (via a genomic relationship matrix) to that population's members, based on multilocus genotype frequencies.

Usage

```
gl.assign.grm(x, unknown, verbose = NULL)
```

Arguments

x	Name of the genlight object containing the SNP data [required].
unknown	Name of the individual to be assigned to a population [required].
verbose	Verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log; 3, progress and results summary; 5, full report [default 2, unless specified using gl.set.verbosity].

Details

This function is a re-implementation of the function multilocus_assignment from package gstudio. Description of the method used in this function can be found at: https://dyerlab.github.io/applied_population_genetics/populat assignment.html

Value

A named `numeric` vector giving, for each candidate population, the mean pairwise GRM (identity-by-descent) value between the unknown individual and that population's members, sorted from most to least related.

Author(s)

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

Examples

```
require("dartR.data")
if (requireNamespace("rrBLUP", quietly = TRUE)) {
  if (isTRUE(getOption("dartR_fbm"))) platypus.gl <- gl.gen2fbm(platypus.gl)
  res <- gl.assign.grm(platypus.gl, unknown = "T27")
}
```

---

gl.diagnostics.relatedness

Measure the bias and accuracy of relatedness estimators on data simulated to match a genlight object

---

Description

Relatedness estimates are biased when a dataset has population structure, inbreeding, or a sample made of families, and the size of the bias depends on the estimator, the relationship and the data. This function measures it for the data at hand. It simulates a population that reproduces the structure, diversity, inbreeding, effective population size, missing data and (optionally) family composition of x, lets it mate at random for a few generations so that every pairwise relationship is known from the pedigree, estimates relatedness with every available estimator on samples like x, and reports each estimator's bias and error against the true kinship, by relationship class. Without a simulation, it compares the estimators on x itself, and against a pedigree attached to x when there is one.

Usage

```
gl.diagnostics.relatedness(
  x,
  cleanup = FALSE,
  ref_variables = NULL,
  sim_variables = NULL,
 which_tests = c("wang", "lynchli", "lynchrd", "ritland", "quellergt", "dyadml"),
  run_sim = FALSE,
  IncludePlots = FALSE,
  plotOut = FALSE,
  varOut = FALSE,
  rmseOut = FALSE,
  numberIterations = 1,
  numberGenerations = 3,
  genToSave = "all",
  run.e9 = FALSE,
  E9Inbreed = FALSE,
  e9Path = NULL,
  verbose = NULL,
  e9parallel = FALSE,
  nCores = 1,
  includedPed = FALSE,
  biasOut = FALSE,
  simMissing = TRUE,
  Ne = NULL,
  neest.path = NULL,
  analysisUnit = c("generation", "pooled"),
  families = NULL,
  colony.path = NULL,
  familyParents = FALSE,
  coancestry.path = NULL
)
```

Arguments

x	A genlight object containing SNP data [required].
cleanup	Logical. Before anything else, keep only loci with no missing data (call rate 1), filter individuals by heterozygosity (gl.filter.heterozygosity defaults) and drop all-NA loci [default FALSE].
ref_variables	Path to the reference variable file for dartR.sim::gl.sim.WF.table [default NULL, which builds one that mirrors x; see Details].
sim_variables	Path to the simulation variable file for dartR.sim::gl.sim.WF.run [default NULL, which builds one that mirrors x; see Details]. A file given here is used as it is.
which_tests	Relatedness estimators of gl.relatedness: any of "wang", "lynchli", "lynchrd", "ritland", "quellergt", "dyadml" and "trioml". rrBLUP (genomic relationship matrix) is always added. "trioml" is left out of the default: with biallelic

SNPs it gives the same estimates as "dyadml" (pairwise correlation 0.999 on platypus.gl, with bias and RMSE within 0.001 by relationship class) at about 150 times the run time (about 25 minutes for 300 individuals) [default all but "trioml"].

`run_sim` Logical. If TRUE, simulate data that mirror x and measure bias against the simulated pedigree [default FALSE].

`IncludePlots` Logical. If TRUE, build and return plots [default FALSE].

`plotOut` Logical. If TRUE, print the plots (requires `IncludePlots = TRUE`) [default FALSE].

`varOut` Logical. If TRUE, return the variance of each estimator by relationship class [default FALSE].

`rmseOut` Logical. If TRUE, return the root mean square error of each estimator against the pedigree kinship, by relationship class [default FALSE].

`numberIterations`
Integer. Number of independent simulations [default 1].

`numberGenerations`
Integer. Number of generations of random mating after the founders; minimum 3 [default 3].

`genToSave` "all", or the positions of the stored generations to keep, generation 0 (the founders) being the first [default "all"].

`run.e9` Logical. If TRUE, also estimate kinship with EMIBD9 [default FALSE].

`E9Inbreed` Logical. If TRUE, run EMIBD9 a second time allowing for inbreeding [default FALSE].

`e9Path` Path to the folder with the EMIBD9 binaries, needed when `run.e9 = TRUE` [default NULL].

`verbose` Verbosity: 0, silent or fatal errors; 1, begin and end and warnings; 2, progress log, including Ne, founder inbreeding and family sizes used; 3, progress and results summary; 5, full report [default 2, unless specified using gl.set.verbosity].

`e9parallel` Logical. Run EMIBD9 in parallel [default FALSE].

`nCores` Integer. Number of cores for EMIBD9 in parallel [default 1].

`includedPed` Logical. If TRUE (and `run_sim = FALSE`), x has a pedigree in `x@other$ind.metrics` (columns id, dad, mom; missing parents 0 or NA) and the estimates of x are compared with it [default FALSE].

`biasOut` Logical. If TRUE, return the bias of each estimator (mean of estimated minus pedigree kinship), by relationship class [default FALSE].

`simMissing` Logical. If TRUE, each simulated individual takes the missing loci of a random individual of x from the same population, so the estimators face the missing data of x [default TRUE].

`Ne` Effective population size of each population of x, one value or one per population in the order of `levels(pop(x))` [default NULL, which estimates it with `dartR.popgen::gl.LDNe` when `neest.path` is given]. Give it whenever you can: LDNe needs larger samples than most relatedness studies have (see Details).

`neest.path` Path to the folder with the NeEstimator binary, used to estimate Ne when `Ne` is NULL [default NULL].

`analysisUnit` "generation" estimates relatedness within each stored generation, a sample the size of x, so the estimators take their allele frequencies from a sample like x; pairs from different generations (parent-offspring, grandparent-grandchild, avuncular) are then estimated only when parents are sampled with their offspring. "pooled" estimates relatedness among all stored generations of an iteration together [default "generation"].

`families` Family structure of x to reproduce in the simulated samples: the name of a column of `x@other$ind.metrics` with a full-sib family identifier for each individual (NA or "" when not in a family), or "colony" to reconstruct full-sib families with `gl.run.colony` (needs `colony.path`) [default NULL, samples of unrelated individuals].

`colony.path` Path to the folder with the COLONY executable, used when `families = "colony"` [default NULL].

`familyParents` Logical. If TRUE, x also contains the parents of its families, and the simulated samples include them; they count towards the sample size [default FALSE].

`coancestry.path`
Path to the folder with the program `dartR_coancestry`, which computes the estimators in `which_tests`; see `gl.relatedness` [required].

Details

What the output means: Every relatedness estimator measures kinship against allele frequencies taken from the sample itself, so its values shift with the composition of the sample: population structure raises estimates within populations and lowers them between populations, inbreeding raises the kinship of relatives beyond what the estimators expect, and a sample of a few families makes unrelated pairs look related. The simulated individuals have known relationships, so the difference between each estimate and the true kinship, averaged by relationship class, is the bias to expect when the same estimator is applied to x. A small bias and RMSE for a class means the estimator can be trusted for that relationship in data like x; the most suitable estimator is the one with the smallest bias and RMSE for the relationships that matter to the question.

How the simulation mirrors x: When `ref_variables` and `sim_variables` are NULL, the variable files are built from the ones shipped with dartR.sim so that the simulated population matches x:

- Loci: the loci of x with their allele frequencies in each population, and no other loci (real_freq, chunk_neutral_loci = 0). x with 100 loci or fewer gets fewer, longer chromosome chunks on the same 1000 cM map, as dartR.sim needs more loci than chunks.
- Population structure: the populations of x (real_pops; x without populations is one population). With two or more populations, founder allele frequencies are shrunk toward their mean to remove the sampling noise of x (real_freq_shrink = "auto"), and migration is set from the FST of x (real_migration); dispersal starts in generation 2.
- Effective population size: each population has the Ne of the matching population of x (ne_phase2) and a census size of 2 x ceiling(max(1.25 Ne, n / 2)), above 2 Ne because parents that mate with several partners halve Ne. There is no default Ne: the function stops when `Ne` is not given and cannot be estimated. `gl.LDNe` estimates are flagged as unreliable when their jackknife

upper limit is infinite or they exceed 10 times the sample size; in tests on five datasets with 9-41 individuals per population, gl.LDNe failed or was flagged for at least one population in every dataset.

- Inbreeding: the founders carry the inbreeding of their population in x (inbreeding_founders), F = 1 - sum(Ho) / sum(uHe) measured on loci called in at least 99% of individuals (the threshold is lowered in steps of 0.01 until 100 loci pass; negative values are set to 0), because heterozygote calls lost at poorly called loci inflate F. Mating is random afterwards (sib_mating_phase2 = 0), so the founders' inbreeding is not passed on to their offspring.
- Mating: parents are drawn with replacement (replace_parents), so an individual can mate with several partners and half sibs occur.
- Samples: every stored generation holds a sample with the sample sizes of x (real_sample_size), including the founders (generation 0, store_founders), and simulated genotypes take the missing data of x (`simMissing`).
- Families: with `families`, every stored generation from generation 1 on is sampled with the family sizes of x per population, and with their parents when `familyParents = TRUE` (sample_families, sample_parents); generation 0, never sampled as families, is then left out of the analysis. Family sizes are reported at verbose >= 2 so they can be checked; kinship estimates are not used to find families, as they carry the bias being measured. Families that share parents (e.g. two sires crossed with two dams) need a variable file with dartR.sim's sample_crosses or sample_design, given in `sim_variables`.

All other variables take the values of the files shipped with dartR.sim. Options that the installed dartR.sim does not know are left out.

The true kinship: All estimates are on the kinship scale (`gl.relatedness` returns relatedness, which is halved). Each pair gets its exact pedigree kinship (column rel) from the full simulated pedigree, including ancestors that were not sampled (dartR.sim store_pedigree), computed recursively with founders unrelated to each other; a founder's kinship with itself is (1 + F) / 2, with F its realised inbreeding. Each pair also gets its closest relationship class (column RelDegree): parent_offspring, full_sibs, half_sibs, grandparent_grandchild, avuncular (aunt or uncle and niece or nephew), full_first_cousins, great_grandparent_grandchild, half_avuncular, half_first_cousins, second_cousins, other_relatives (pedigree kinship above 0 but none of these classes) or unrelated (pedigree kinship 0). Bias is the mean of estimate minus rel and RMSE the root mean square of the same differences, by class.

What is not simulated: Genotyping error is not simulated: missing data are copied from x, but lost heterozygote calls, allelic dropout and paralogous loci (loci heterozygous in nearly every individual) are not, so for data with these problems the bias is under-reported; remove such loci before running the function. Loci are placed on a simulated map, so the linkage between the loci of x is not reproduced. Substructure within the populations of x (e.g. sites within a lineage) is reproduced only when the sites are given as the populations of x.

Requirements: The estimators in `which_tests` are computed by `gl.relatedness`, which needs the separately downloaded program `dartR_coancestry` (argument `coancestry.path`; see `gl.relatedness`). The simulation needs a recent dartR.sim (store_founders, store_pedigree, ne_phase2, inbreeding_founders, sample_families). Estimating Ne needs dartR.popgen and NeEstimator; reconstructing families needs COLONY; the EMIBD9 estimates need EMIBD9. A run of 2 iterations on 81 individuals and 557 loci takes about a minute.

Value

An S4 object with the slots:

- @InputDf: x, after filtering when `cleanup = TRUE`.
- @SimOutput: one genlight per iteration with all stored generations; `@other$ind.metrics` holds each individual's parents, generation and sampling role, and `@other$sim.vars` the simulation variables, including values derived from x (e.g. ne_expected, migrants_real, freq_shrink_lambda).
- @MergedDf: one data frame per iteration, one row per pair, with the estimates of each estimator; with a pedigree, also RelDegree and rel; with a simulation, the generation of the pair.
- @corOutList: tables of bias, RMSE and variance by relationship class (`biasOut`, `rmseOut`, `varOut`).
- @corVals: correlations between estimators and with rel.
- @plotList: plots per iteration; with a pedigree, estimates by relationship class (boxplots and densities) and a tile plot of the bias and RMSE of each estimator by class.

Author(s)

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

See Also

gl.relatedness, gl.run.colony, gl.sim.WF.run, gl.filter.callrate

Other captive management: gl.parentage.power(), gl.plot.network(), gl.run.EMIBD9(), gl.run.colony(), gl.sim.relatedness(), gl2colony()

Examples

```
## Not run:
# requires the program dartR_coancestry (see gl.relatedness)
dartRverse::gl.download.binary(software = "coancestry", out.dir = tempdir())
x <- gl.filter.callrate(platypus.gl, threshold = 0.9)
x <- gl.filter.monomorphs(x)
res <- gl.diagnostics.relatedness(x, run_sim = TRUE,
                                  coancestry.path = file.path(tempdir(),
                                                              "coancestry"),
                                  numberIterations = 2,
                                  Ne = c(40, 40, 90),
                                  biasOut = TRUE, rmseOut = TRUE,
                                  IncludePlots = TRUE)
res@corOutList@biasPlot   # bias (RMSE) by estimator and class
res@plotList[[1]][[3]]     # the same as a tile plot

# sibs: a genlight made of full-sib families, whose families are
# reconstructed with COLONY
res2 <- gl.diagnostics.relatedness(sibs, run_sim = TRUE, Ne = 30,
                                   families = "colony",
                                   colony.path = "~/programs",
```

```
                                    biasOut = TRUE)

## End(Not run)
```

`gl.filter.parent.offspring` *Filters putative parent offspring within a population*

Description

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

Usage

```
gl.filter.parent.offspring(
  x,
  min.rdepth = 12,
  min.reproducibility = 1,
  range = 1.5,
  method = "best",
  rm.monomorphs = FALSE,
  plot_theme = theme_dartR(),
  plot_colors = NULL,
  plot.file = NULL,
  plot.dir = NULL,
  verbose = NULL
)
```

Arguments

`x`	Name of the genlight object containing the SNP genotypes [required].
`min.rdepth`	Minimum read depth to include in analysis [default 12].
`min.reproducibility`	Minimum reproducibility to include in analysis [default 1].
`range`	Specifies the range to extend beyond the interquartile range for delimiting outliers [default 1.5 interquartile ranges].
`method`	Method of selecting the individual to remove from each pair in a parent offspring relationship: 'best' removes the individual with more missing genotypes (lower call rate) and keeps the other; 'random' removes one at random [default 'best'].

rm.monomorphs	If TRUE, remove monomorphic loci after filtering individuals [default FALSE].
plot_theme	Theme for the plot. See Details for options [default theme_dartR()].
plot_colors	List of two color names for the borders and fill of the plots [default NULL, which uses gl.colors(2)].
plot.file	Name for the RDS binary file to save (base name only, exclude extension) [default NULL]
plot.dir	Directory to save the plot RDS files [default as specified by the global working directory or tempdir()]
verbose	Verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log ; 3, progress and results summary; 5, full report [default 2, unless specified using gl.set.verbosity].

Details

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

To reduce the frequency of miscalls, and so emphasize the difference between true parent-offspring pairs and unrelated pairs, the data can be filtered on read depth. Typically minimum read depth is set to 5x, but you can examine the distribution of read depths with the function gl.report.rdepth and push this up with an acceptable loss of loci. 12x might be a good minimum for this particular analysis. It is sensible also to push the minimum reproducibility up to 1, if that does not result in an unacceptable loss of loci. Reproducibility is stored in the slot @other$loc.metrics$RepAvg and is defined as the proportion of technical replicate assay pairs for which the marker score is consistent. You can examine the distribution of reproducibility with the function gl.report.reproducibility.

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

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

Note that if your dataset does not contain RepAvg or rdepth among the locus metrics, the filters for reproducibility and read depth are not used.

The pairs are identified with gl.report.parent.offspring using the same arguments. An individual that belongs to several pairs is removed first, because that resolves all of its pairs at once; the remaining pairs are resolved in order of evidence (lowest proportion of inconsistent loci first), and a pair drops out as soon as either member has been removed. This keeps the number of removed individuals low.

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

The genlight object without one individual from each putative parent-offspring pair. If no pairs are found, the object is returned unchanged.

Author(s)

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

See Also

gl.report.rdepth , gl.report.reproducibility, gl.report.parent.offspring

Examples

```
if (isTRUE(getOption("dartR_fbm"))) testset.gl <- gl.gen2fbm(testset.gl)
out <- gl.filter.parent.offspring(testset.gl[1:10, 1:50])
```

---

gl.grm *Calculates an identity by descent matrix*

Description

This function calculates the mean probability of identity by state (IBS) across loci that would result from all the possible crosses of the individuals analyzed. IBD is calculated by an additive relationship matrix approach developed by Endelman and Jannink (2012) as implemented in the function A.mat (package rrBLUP).

Usage

```
gl.grm(
  x,
  plotheatmap = TRUE,
  palette_discrete = NULL,
  palette_convergent = NULL,
  legendx = 0,
  legendy = 0.5,
  label.size = 0.75,
  legend.title = "Populations",
  plot.file = NULL,
  plot.dir = NULL,
  verbose = NULL,
  ...
)
```

Arguments

x	Name of the genlight object containing the SNP data [required].
plotheatmap	A switch if a heatmap should be shown [default TRUE].
palette_discrete	the color of populations [gl.select.colors].
palette_convergent	A convergent palette for the IBD values [default convergent_palette].
legendx	x coordinates for the legend[default 0].
legendy	y coordinates for the legend[default 0.5].
label.size	Specify the size of the population labels [default 0.75].
legend.title	Legend title [default "Populations"].
plot.file	Name for the RDS binary file to save (base name only, exclude extension); saved only when plotheatmap = TRUE [default NULL]
plot.dir	Directory in which to save files [default = working directory]
verbose	Verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log ; 3, progress and results summary; 5, full report [default 2 or as specified using gl.set.verbosity].
...	Parameters passed to function A.mat from package rrBLUP. Unless min.MAF is given, it is set to 1/(2n) - 1e-10, so loci with a single copy of the minor allele are kept on every platform.

Details

This function uses the A.mat function from the rrBLUP package. This method follows the approach developed by Endelman and Jannink (2012).

Two alleles are Identical by State (IBS) if they are the same in state, regardless of whether they come from a common ancestor. Two alleles are Identical by Descent (IBD) if they are inherited from a common ancestor. While IBS does not necessarily imply IBD, using high-density SNP data improves the estimation of IBD probabilities from IBS measures.

The matrix is on the relatedness scale: each diagonal element estimates 1+f, where f is the inbreeding coefficient of the individual, and each off-diagonal element estimates twice the kinship of the pair. Both are relative to the allele frequencies of the supplied individuals, so f can be negative (diagonal below 1), and the matrix sums to zero: the mean of the off-diagonal elements is -(mean of the diagonal)/(n-1), where n is the number of individuals. Kinship is half these values.

This function also plots a heatmap, and a dendrogram, of the matrix. Individual names are shown in the margins of the heatmap and colors represent different populations.

A file-backed (FBM) object is not decoded as a whole for A.mat's default method (mean imputation, no shrinkage; min.MAF and max.missing may be given): the matrix is computed in blocks of loci (dartR.base::utils.fbm.grm) and agrees with A.mat to about 1e-13. Other A.mat options (shrink, impute.method = "EM", return.imputed) need the whole matrix: the object is then converted to a genlight, with a warning stating the memory A.mat uses (about four doubles per genotype). Gen-backed objects use A.mat as before.

Value

A square genomic relationship matrix on the relatedness scale (diagonal 1+f, off-diagonal twice the kinship), with row and column names set to the individual names and attr(, "scale") = "relatedness", returned invisibly.

Author(s)

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

References

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

See Also

gl.grm.network

Other inbreeding functions: gl.grm.network(), gl.inbreeding()

Examples

```
if (isTRUE(getOption("dartR_fbm"))) platypus.gl <- gl.gen2fbm(platypus.gl)
gl.grm(platypus.gl[1:10, 1:100])
```

---

gl.grm.network *Represents a similarity matrix as a network*

Description

This script takes a genomic relationship matrix and represents the kinship among the specimens as a network diagram.

Usage

```
gl.grm.network(
  G,
  x,
  standardise = FALSE,
  categorise = FALSE,
  color.categories = c("#E63E94", "#E5D44C", "#3ED2E6"),
  method = "fr",
  node.size = 8,
  node.label = TRUE,
```

```
  node.label.size = 2,
  node.label.color = "black",
  link.color = NULL,
  link.size = 2,
  kinship.threshold = 0.125,
  title = "Network of a similarity matrix",
  legend.title = "Populations",
  title.size = 16,
  legend.size = 14,
  palette_discrete = NULL,
  plot.dir = NULL,
  plot.file = NULL,
  verbose = NULL
)
```

Arguments

G	A square relationship matrix, such as the output of gl.grm or the $rel element of gl.run.EMIBD9. Row and column names must be the individual names of x. The scale is read from attr(G, "scale"): a G tagged "kinship" is used as kinship directly; any other G (tagged "relatedness", or untagged) is taken as relatedness and halved [required].
x	A genlight object from which the matrix was generated [required].
standardise	Whether to standardise matrix using Goudet et al method, see details [default FALSE].
categorise	Whether to categorise the color of the link representing kinship values (G / 2) into relationships. Same Individual (>=0.3), Full Siblings / Parent-Offspring (>=0.2 & <0.3) and Half Siblings (>=0.1 & <0.2) [default FALSE].
color.categories	A vector of three colors for the above kinship categories, in the order Same Individual, Full Siblings / Parent-Offspring, Half Siblings [default = c("#E63E94","#E5D44C","#3ED2E6")].
method	One of 'fr', 'kk', 'gh' or 'mds' [default 'fr'].
node.size	Size of the symbols for the network nodes [default 8].
node.label	TRUE to display node labels [default TRUE].
node.label.size	Size of the node labels [default 2].
node.label.color	Color of the text of the node labels [default 'black'].
link.color	Colors for links, either a vector of colors or a color palette function [NULL].
link.size	Size of the links [default 2].
kinship.threshold	Threshold of kinship value (G / 2) to display in the network diagram [default 0.125].
title	Title for the plot [default 'Network of a similarity matrix'].
legend.title	Title for the legend [default "Populations"].

`title.size`	Font size of the title [default 16].
`legend.size`	Font size of the legend [default 14].
`palette_discrete`	A discrete set of colors with as many colors as there are populations in the dataset [default NULL].
`plot.dir`	Directory to save the plot RDS files [default as specified by the global working directory or tempdir]
`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, progress log ; 3, progress and results summary; 5, full report [default 2 or as specified using gl.set.verbosity].

Details

The gl.grm.network function creates a network diagram that represents genetic relationships among individuals in a dataset. Off-diagonal elements of G are relatedness coefficients, so they are divided by 2 to obtain the kinship values (not for G tagged as kinship, see G) that are compared with kinship.threshold, used for the categories and returned in the kinship matrix.

Layout options

Four layout options are implemented in this function:

- 'fr' Fruchterman-Reingold layout layout_with_fr (package igraph)
- 'kk' Kamada-Kawai layout layout_with_kk (package igraph)
- 'gh' Graphopt layout layout_with_graphopt (package igraph)
- 'mds' Multidimensional scaling layout layout_with_mds (package igraph)

Standardise matrix using Goudet et al method

Choosing meaningful thresholds to represent relationships between individuals can be challenging because kinship and inbreeding coefficients are relative measures. To standardize a genomic relationship matrix (GRM), such as the one produced by the function gl.grm, and facilitate interpretation, the function re-expresses kinship relative to the average pair of individuals in the sample, following Goudet et al. (2018):

1. Off-diagonal elements are converted to kinship theta (G / 2, or G itself when G is tagged as kinship).

2. The mean kinship over all pairs of distinct individuals, mean_theta, is taken as the reference.

3. Each kinship becomes (theta - mean_theta) / (1 - mean_theta), so the average pair has kinship 0 and a pair identical by descent at every locus keeps kinship 0.5.

For gl.grm input, which is already centred on the sample allele frequencies, mean_theta is close to 0 and the change is small; it matters for matrices on another reference, such as EMIBD9 output. The diagonal (inbreeding) is not used, because the mean inbreeding of the sample reflects population structure and missing data rather than the relationship between pairs. Because the average pair becomes the zero point, a sample made up mostly of relatives gives standardised values below the pedigree expectations (e.g. parent-offspring 0.12 instead of 0.21 in an EMIBD9 run on a single captive family with a few wild individuals); standardise only when the sample represents the wider population.

Below is a table modified from Speed & Balding (2015) showing kinship values, and their confidence intervals (CI), for different relationships that could be used to guide the choosing of the kinship threshold in the function.

Relationship	**Kinship**	**95% CI**
Identical twins / clones / same individual	0.5	-
Sibling / Parent-Offspring	0.25	(0.204, 0.296)
Half-sibling	0.125	(0.092, 0.158)
First cousin	0.062	(0.038, 0.089)
Half-cousin	0.031	(0.012, 0.055)
Second cousin	0.016	(0.004, 0.031)
Half-second cousin	0.008	(0.001, 0.020)
Third cousin	0.004	(0.000, 0.012)
Unrelated	0	-

Value

Invisibly, a named list:

- plot – the network plot (ggplot object).
- kinship – a matrix of pairwise kinship values (G / 2, or the standardised values when standardise = TRUE). Only the lower triangle is filled; the upper triangle is NA and the diagonal is set to 0. Rows and columns follow the row order of G.

Author(s)

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

References

- Endelman, J. B. , Jannink, J.-L. (2012). Shrinkage estimation of the realized relationship matrix. G3: Genes, Genomics, Genetics 2, 1405.
- Goudet, J., Kay, T., & Weir, B. S. (2018). How to estimate kinship. Molecular Ecology, 27(20), 4121-4135.
- Speed, D., & Balding, D. J. (2015). Relatedness in the post-genomic era: is it still useful?. Nature Reviews Genetics, 16(1), 33-44.

See Also

gl.grm

Other inbreeding functions: gl.grm(), gl.inbreeding()

Examples

```
if (requireNamespace("igraph", quietly = TRUE) & requireNamespace("rrBLUP",
  quietly = TRUE
) & requireNamespace("fields", quietly = TRUE)) {
```

```
  if (isTRUE(getOption("dartR_fbm"))) possums.gl <- gl.gen2fbm(possums.gl)
  t1 <- possums.gl
  # filtering on call rate
  t1 <- gl.filter.callrate(t1)
  t1 <- gl.subsample.loc(t1, n = 100)
  # relatedness matrix
  res <- gl.grm(t1, plotheatmap = FALSE)
  # relatedness network
  res2 <- gl.grm.network(res, t1, kinship.threshold = 0.125)
}
```

`gl.inbreeding` *Estimates individual inbreeding coefficients with PLINK and GCTA*

Description

Estimates the inbreeding coefficient of each individual from SNP data by running PLINK 1.9 (–het) and GCTA (–ibc), and plots the estimates of each population, individuals sorted by value.

Usage

```
gl.inbreeding(
  x,
  method = c("plink", "gcta"),
  by.pop = FALSE,
  plink.path = getwd(),
  gcta.path = getwd(),
  plot.display = TRUE,
  plot.theme = theme_dartR(),
  plot.colors = NULL,
  ind.labels = FALSE,
  label.size = 6,
  plot.file = NULL,
  plot.dir = NULL,
  verbose = NULL
)
```

Arguments

`x`	Name of the genlight object containing the SNP data [required].
`method`	Programs to run: "plink", "gcta" or both [default c("plink", "gcta")].
`by.pop`	If TRUE, each population is analysed on its own, so allele frequencies come from that population; if FALSE, from all individuals pooled. See Details [default FALSE].
`plink.path`	Path to the folder holding the PLINK 1.9 executable (plink, or plink.exe on Windows), or the path to the executable itself. PLINK is needed for both methods [default getwd()].

gcta.path	Path to the folder holding the GCTA executable (gcta64, or gcta64.exe on Windows), or the path to the executable itself. Only used when method includes "gcta" [default getwd()].
plot.display	If TRUE, the plot is displayed in the plot window [default TRUE].
plot.theme	Theme for the plot. See Details for options [default theme_dartR()].
plot.colors	A vector of colours, one per population (named by population, or in the order of the population levels), or a palette function [default gl.select.colors(x)].
ind.labels	If TRUE, the names of the individuals are shown on the vertical axis of each panel [default FALSE].
label.size	Size, in points, of the names of the individuals [default 6].
plot.file	Name for the RDS binary file to save (base name only, exclude extension) [default NULL].
plot.dir	Directory to save the plot RDS files [default as specified by the global working directory or tempdir()].
verbose	Verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log; 3, progress and results summary; 5, full report [default 2, unless specified using gl.set.verbosity].

Details

Both programs measure the excess of homozygosity of each individual over the homozygosity expected from allele frequencies estimated in the sample. PLINK –het reports F = (O(HOM) - E(HOM)) / (N(NM) - E(HOM)), where O(HOM) is the observed and E(HOM) the expected number of homozygous loci, and N(NM) the number of loci called in the individual. GCTA –ibc reports three estimators (Yang et al. 2011), with x the allele count (0, 1, 2) at a locus, p the allele frequency and h = 2p(1 - p), each averaged over the loci called in the individual: Fhat1 = (x - 2p)^2 / h - 1, the diagonal of the genomic relationship matrix minus 1; Fhat2 = 1 - x(2 - x) / h, based on excess homozygosity; and Fhat3 = (x^2 - (1 + 2p)x + 2p^2) / h, based on the correlation between uniting gametes. Fhat3 gives a large weight to rare alleles: an individual homozygous for an allele at frequency p adds (1 - p) / p at that locus, so a few such loci can push Fhat3 well above 1. Removing rare alleles beforehand (gl.filter.maf) reduces this effect.

Loci that are monomorphic or not called in the individuals analysed are removed before the programs run, so PLINK and GCTA estimate on the same loci (GCTA would otherwise count monomorphic loci in its denominator and pull every estimate towards 0). All loci are treated as autosomal and placed on one chromosome; neither estimator uses map positions. Remove sex-linked loci first (for example with dartR.sexlinked gl.drop.sexlinked), because loci on the X chromosome are hemizygous in males and would make males appear more inbred.

With by.pop = FALSE and more than one population, the expected homozygosity comes from the pooled allele frequencies, so the differentiation between populations (the Wahlund effect) is counted as inbreeding in every individual. With by.pop = TRUE, each population is run separately: populations of one individual get NA (every homozygous locus is monomorphic, so F would be -1 by construction), and populations of fewer than 10 individuals get a warning that their allele frequencies, and hence the estimates, are imprecise.

PLINK 1.9 can be downloaded from https://www.cog-genomics.org/plink/ and GCTA from https://yanglab.westlake.edu.cn/software/gcta/. The files for both programs are written to a new folder under tempdir(), which is deleted on exit.

The plot shows PLINK F and GCTA Fhat3 side by side, one row of panels per population, with the population names on the right. Each panel sorts the individuals by its own estimate, highest at the bottom, so with ind.labels = TRUE the two panels list the names in different orders. Names need about 1 mm of height each: save a plot with many individuals with a larger height, or lower label.size. A colour vector can be obtained with gl.select.colors() and passed with the plot.colors parameter. Themes can be obtained from https://ggplot2.tidyverse.org/reference/ggtheme.html; the panel borders, grid lines, population strips and titles are drawn over the theme. If a plot.file is given, the ggplot 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().

Value

A data frame, returned invisibly, with one row per individual in the order of x: id, pop, and for PLINK the columns O.HOM, E.HOM, N.NM and F, and for GCTA the columns NOMISS (loci used), Fhat1, Fhat2 and Fhat3. Individuals that could not be estimated have NA.

Author(s)

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

References

- Chang, C. C., Chow, C. C., Tellier, L. C., Vattikuti, S., Purcell, S. M., & Lee, J. J. (2015). Second-generation PLINK: rising to the challenge of larger and richer datasets. GigaScience, 4(1), 7.
- Purcell, S., Neale, B., Todd-Brown, K., Thomas, L., Ferreira, M. A., Bender, D., ... & Sham, P. C. (2007). PLINK: a tool set for whole-genome association and population-based linkage analyses. The American Journal of Human Genetics, 81(3), 559-575.
- Yang, J., Lee, S. H., Goddard, M. E., & Visscher, P. M. (2011). GCTA: a tool for genome-wide complex trait analysis. The American Journal of Human Genetics, 88(1), 76-82.

See Also

gl.relatedness, gl.run.EMIBD9

Other inbreeding functions: gl.grm(), gl.grm.network()

Examples

```
## Not run:
# needs PLINK 1.9 and GCTA installed; point to their folders
res <- gl.inbreeding(platypus.gl, plink.path = "~/programs",
                     gcta.path = "~/programs")
# allele frequencies from each population
res.pop <- gl.inbreeding(platypus.gl, by.pop = TRUE,
                         plink.path = "~/programs",
                         gcta.path = "~/programs")
# names of the individuals on the vertical axis
```

```
res <- gl.inbreeding(platypus.gl, ind.labels = TRUE,
                     plink.path = "~/programs",
                     gcta.path = "~/programs")
# PLINK only
res.plink <- gl.inbreeding(platypus.gl, method = "plink",
                           plink.path = "~/programs")

## End(Not run)
```

`gl.parentage.power` *Estimate the power of a SNP panel to assign parentage*

Description

Quantifies, by simulation, how reliably a SNP panel can assign offspring to their parents given a set of candidate parents. Offspring are simulated by Mendelian inheritance from parent pairs drawn from the supplied genlight, optionally degraded with genotyping error and missing data to mimic a real DArTseq run. Each simulated offspring is then compared against every candidate parent by counting opposite-homozygote mismatches (a candidate that shares no allele at a locus cannot be a parent there), and the true parent's retention, the rate of uniquely correct assignment, and the rate of false compatibility with non-parents are tallied. The whole procedure is repeated across a rarefaction series of locus counts, giving a power-versus- number-of-loci curve.

Usage

```
gl.parentage.power(
  x,
  n.offspring = 200,
  engine = "exclusion",
  pairs = TRUE,
  error.rate = 0,
  missing.rate = 0,
  max.mismatch = 0,
  n.loci.steps = NULL,
  n.rep = 1,
  plot.display = TRUE,
  plot.theme = theme_dartR(),
  plot.colors = NULL,
  plot.file = NULL,
  plot.dir = NULL,
  verbose = NULL,
  ...
)
```

Arguments

x	Name of the genlight object holding the candidate parents [required].
n.offspring	Number of offspring to simulate per replicate [default 200].
engine	Assignment engine: "exclusion" for the fast Mendelian opposite-homozygote / trio test used to build the power curve, or "colony" to assign the simulated cohort with COLONY via gl.run.colony for verification against the exclusion result [default "exclusion"].
pairs	If TRUE, assess parent-pair (trio) power: both parents of each offspring are among the candidates, and the offspring must be consistent with one allele from each of the two. If FALSE, assess single-parent power: only one parent of each offspring was sampled (the other parent is removed from the candidates), and candidates are excluded by opposite-homozygote mismatches [default TRUE].
error.rate	Per-genotype rate at which a call is replaced by a random genotype, mimicking miscalls [default 0].
missing.rate	Per-genotype rate at which a call is set to missing [default 0].
max.mismatch	Maximum number of opposite-homozygote mismatches at which a candidate is still treated as a compatible parent. Must be raised above 0 whenever error.rate or missing.rate is non-zero, or the true parent will be wrongly excluded [default 0].
n.loci.steps	Vector of locus counts at which to evaluate power. NULL builds an even series from a floor up to all polymorphic loci [default NULL].
n.rep	Number of independent simulation replicates to average over. With engine = "colony" each replicate simulates a new cohort and is a separate COLONY run [default 1].
plot.display	If TRUE, the power curve is displayed [default TRUE].
plot.theme	Theme for the plot [default theme_dartR()].
plot.colors	List of two color names for the two power lines [default c("#2171B5","#6BAED6")].
plot.file	Name for the RDS binary file to save (base name only) [default NULL].
plot.dir	Directory to save plot RDS files [default tempdir()].
verbose	Verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log; 3, progress and results summary; 5, full report [default NULL, unless specified using gl.set.verbosity].
...	When engine = "colony", additional parameters passed to gl.run.colony (e.g. colony.path, length.run, num.runs, likelihood).

Details

The function assumes the candidate panel is a single breeding population, as a hatchery broodstock normally is: parents are drawn from it and every offspring is tested against it, so it should be representative of the real breeding group. Only loci polymorphic in the panel carry information, and monomorphic loci are dropped before simulation.

Offspring are generated one true parent pair at a time by transmitting one allele from each parent at each locus (a parent scored as a missing genotype transmits a missing genotype).

With `engine = "exclusion"` (the default) assignment uses a fast Mendelian test with no external program: in pair mode an offspring must be consistent with one allele from each of the two candidates (the trio test); in single-parent mode a candidate is excluded only by an opposite-homozygote mismatch. In single-parent mode one of the two true parents of each offspring, chosen at random, is the sampled parent, and the other is removed from the candidates, as when the second parent was never genotyped. If both stayed in, they would tie at zero mismatches whenever there is no genotyping error, and no offspring could be assigned to one of them alone. The exclusion engine is run across a rarefaction series of locus counts to give the power curve.

Three quantities are reported at each locus count:

- **true.retained** – the proportion of offspring whose sampled parent (single mode) or true pair (pair mode) is still within the mismatch tolerance (sensitivity);
- **correct.unique** – the proportion of offspring assigned uniquely and correctly. In pair mode, the true pair is within the tolerance and no other pair of candidates is; every pair is checked, not a sample. In single mode, the sampled parent is within the tolerance and has fewer mismatches than every other candidate. It never exceeds true.retained;
- **false.rate** – the per-comparison false-positive rate: the proportion of non-parent candidates (single mode) or of all candidate pairs other than the true pair (pair mode) wrongly within the mismatch tolerance. Being a rate, it does not depend on the size of the candidate pool.

Checking every pair stays fast because a pair can only be compatible when each of its members is: an opposite-homozygote mismatch with one parent is a trio mismatch whatever the other parent's genotype. Only pairs among the candidates that pass the single-parent test are evaluated.

With `engine = "colony"` the simulated cohort is instead assigned by COLONY through `gl.run.colony`, at the full panel, so its accuracy can be checked against the exclusion prediction; this is much slower and needs the COLONY executable. Each replicate is a separate COLONY run in its own folder, `plot.dir/colony_run/rep1`, `rep2` and so on. COLONY assigns parents rather than testing compatibility, and `pairs` is not used, so the three columns mean: true.retained, at least one true parent is among the assigned candidates; correct.unique, the assigned candidates are exactly the two true parents; false.rate, the proportion of offspring assigned at least one non-parent (an assignment error rate, not a per-comparison rate).

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 with `$power` (the power table), `$settings` (the call parameters) and `$plot` (the ggplot object), returned invisibly.

Author(s)

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

See Also

Other captive management: `gl.diagnostics.relatedness()`, `gl.plot.network()`, `gl.run.EMIBD9()`, `gl.run.colony()`, `gl.sim.relatedness()`, `gl2colony()`

Examples

```
# A single breeding population from the example set.
pop1 <- gl.keep.pop(testset.gl, pop.list = popNames(testset.gl)[1],
                    verbose = 0)
out <- gl.parentage.power(pop1, n.offspring = 50, verbose = 0)
```

---

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

Description

This script takes a similarity matrix (such as the output of gl.grm) or a distance matrix (such as the output of dist() or gl.dist.ind()) and represents the relationship among the specimens as a network diagram. In order to use this script, a decision is required on a threshold for relatedness to be represented as link in the network, and on the layout used to create the diagram.

The threshold for relatedness to be represented as a link in the network is specified as the proportion alpha of all pairs to draw: for a similarity matrix the alpha proportion of pairs with the highest values, for a distance matrix the alpha proportion with the lowest values. Often you are looking for relatedness outliers in comparison with the overall relatedness among individuals, so a very conservative value is used (e.g. 0.004), but ultimately, this decision is made as a matter of trial and error. One way to approach this trial and error is to try to achieve a sparse set of links between unrelated 'background' individuals so that the stronger links are preferentially shown.

Link widths are proportional to the strength of the relationship (highest similarity or lowest distance widest), rescaled to between 0.5 and 4.

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

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

Colors of node symbols are those of the rainbow.

Usage

```
gl.plot.network(
  D,
  x = NULL,
  method = "fr",
  node.size = 3,
  node.label = FALSE,
  node.label.size = 0.7,
  node.label.color = "black",
```

```
  alpha = 0.005,
  title = "Network based on genetic distance",
  type = "similarity",
  verbose = NULL
)
```

Arguments

D	A similarity matrix (e.g. from gl.grm) or a distance matrix (e.g. from dist() or gl.dist.ind()); set type accordingly. Row and column names must be the individual names of x when x is given [required].
x	A genlight object from which the D matrix was generated; used to colour nodes by population [default NULL].
method	One of "fr", "kk" or "drl" [default "fr"].
node.size	Size of the symbols for the network nodes [default 3].
node.label	TRUE to display node labels [default FALSE].
node.label.size	Size of the node labels [default 0.7].
node.label.color	Color of the text of the node labels [default 'black'].
alpha	Proportion of all pairs drawn as links: those with the highest similarity, or the lowest distance [default 0.005].
title	Title for the plot [default "Network based on genetic distance"].
type	Whether D holds similarities ("similarity", high values = closely related, e.g. gl.grm) or distances ("distance", low values = closely related, e.g. dist) [default "similarity"].
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

Invisibly, the network that was drawn (an igraph object with the drawn links; the edge attribute weight holds the value from D).

Author(s)

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

See Also

Other captive management: gl.diagnostics.relatedness(), gl.parentage.power(), gl.run.EMIBD9(), gl.run.colony(), gl.sim.relatedness(), gl2colony()

Examples

```
if ((requireNamespace("rrBLUP", quietly = TRUE)) & (requireNamespace("gplots", quietly = TRUE))) {
  if (isTRUE(getOption("dartR_fbm"))) platypus.gl <- gl.gen2fbm(platypus.gl)
  test <- gl.subsample.loc(platypus.gl, n = 100)
  test <- gl.keep.ind(test, ind.list = indNames(test)[1:10])
  D <- gl.grm(test, legendx = 0.04)
  gl.plot.network(D, test)
}
```

---

gl.relatedness *Pairwise relatedness and inbreeding from SNP data (Coancestry estimators)*

Description

Estimates pairwise relatedness (and optionally inbreeding) between individuals in a genlight object using Jinliang Wang's Coancestry estimators. Method-of-moments estimators (Wang, Lynch & Li, Lynch & Ritland, Ritland, Queller & Goodnight) and maximum-likelihood estimators (DyadML, TrioML), plus per-individual inbreeding, are available.

The estimators are computed by the closed-source program `dartR_coancestry`, a C++ translation of Coancestry, which must be downloaded separately (see Details).

Usage

```
gl.relatedness(
  x,
  estimators = c("wang", "lynchli", "lynchrd", "ritland", "quellergt"),
  allow.inbreeding = FALSE,
  n.boots = 0,
  num.trios = 100,
  rng.seed = 42,
  n.threads = 1,
  coancestry.path = getwd(),
  plot.stat = NULL,
  plot.out = TRUE,
  plot.colors = NULL,
  plot.dir = NULL,
  plot.file = NULL,
  verbose = NULL
)
```

Arguments

`x`	Name of the genlight object containing the SNP data [required].
`estimators`	Character vector of estimators; any subset of `"wang"`, `"lynchli"`, `"lynchrd"`, `"ritland"`, `"quellergt"`, `"inbreeding"`, `"dyadml"`, `"trioml"` [default the five moment estimators].

`allow.inbreeding`
: If TRUE, fit the inbreeding-aware ML model for `dyadml`/`trioml` [default FALSE].

`n.boots`
: Bootstrap-over-loci replicates for 95% CIs; 0 disables CIs [default 0].

`num.trios`
: Reference trios for `trioml` [default 100].

`rng.seed`
: Random seed (matters for `dyadml`/`trioml` and the bootstrap) [default 42].

`n.threads`
: Threads for the moment tier (the ML tier is serial) [default 1].

`coancestry.path`
: Path to the folder with the program `dartR_coancestry` (`dartR_coancestry.exe` on Windows); see Details [default getwd()].

`plot.stat`
: Estimator whose matrix is drawn as a heatmap; NULL = the first requested dyad estimator [default NULL].

`plot.out`
: If TRUE, display a heatmap of `plot.stat`; the heatmap is saved when `plot.file` is given, whether or not it is displayed [default TRUE].

`plot.colors`
: Divergent palette; NULL uses `gl.colors("div")` [default NULL].

`plot.dir`
: Directory to save the plot RDS [default working dir or tempdir].

`plot.file`
: Filename (no extension) for the saved RDS plot [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

`as.matrix(x)` supplies the individual x locus alternate-allele dosage matrix (0/1/2, NA missing) the engine expects; allele frequencies are computed internally. When `n.boots > 0`, monomorphic / all-missing loci and all-missing individuals are dropped first (reported at `verbose >= 1`): the bootstrap loop, faithful to Coancestry, does not terminate on a pair with no usable polymorphic locus.

The program `dartR_coancestry` is closed-source and is not part of this package. Download the build for your operating system with

```
dartRverse::gl.download.binary(software = "coancestry",
  out.dir = tempdir())
```

and give its folder in `coancestry.path`, for example `coancestry.path = file.path(tempdir(), "coancestry")`. The genotypes and the results pass between R and the program through temporary files, which are removed afterwards. A run can be stopped as any R command (RStudio's Stop button, Esc or Ctrl-C); the program is then ended and its files removed.

A pair of individuals that shares no called locus has no information on relatedness; every estimate for it is set to NA (the engine returns 0 for the moment estimators and NaN for the likelihood estimators), with a warning at `verbose >= 1`. Individuals dropped by the bootstrap pre-filter keep their rows and columns in the matrices, filled with NA.

The matrices are on the relatedness scale (twice kinship) and carry `attr(, "scale") = "relatedness"`; kinship is half these values.

Agreement with Coancestry: on 30 platypus (platypus.gl, 420 loci with no missing data), `lynchrd`, `ritland` and `quellergt` match `related::coancestry` (the Coancestry Fortran code) within its 4-decimal rounding. `wang` is on average 0.047 higher and `lynchli` 0.018 higher. In simulated biallelic

SNP data the gap for unrelated pairs shrinks with sample size (0.050 at 20 individuals, 0.023 at 50, 0.013 at 120), which points to a different small-sample correction. In the same simulations neither implementation of `wang` or `lynchli` recovers r = 0 for unrelated pairs (both about -0.16), while `quellergt` does (-0.009 at 120 individuals); for full sibs all give 0.57-0.58.

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

Invisibly, a named list: one symmetric individual x individual matrix per requested dyad estimator (e.g. `$wang`; rownames/colnames = `indNames(x)`, diagonal NA, `attr(, "scale") = "relatedness"`); a long per-pair data frame `$dyads` (with `_lo`/`_hi` when `n.boots > 0`); and, when requested, `$delta19` (DyadML deltas), `$trio_delta` (TrioML deltas) and `$inbreeding` (per-individual LH/LR and/or ML inbreeding).

Author(s)

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

References

- Wang, J. (2011). COANCESTRY: a program for simulating, estimating and analysing relatedness and inbreeding coefficients. Molecular Ecology Resources, 11, 141-145.
- Li, C.C. & Horvitz, D.G. (1953). Some methods of estimating the inbreeding coefficient. American Journal of Human Genetics, 5, 107-117.
- Li, C.C., Weeks, D.E. & Chakravarti, A. (1993). Similarity of DNA fingerprints due to chance and relatedness. Human Heredity, 43, 45-52.
- Lynch, M. (1988). Estimation of relatedness by DNA fingerprinting. Molecular Biology and Evolution, 5, 584-599.
- Lynch, M. & Ritland, K. (1999). Estimation of pairwise relatedness with molecular markers. Genetics, 152, 1753-1766.
- Milligan, B.G. (2003). Maximum-likelihood estimation of relatedness. Genetics, 163, 1153-1167.
- Queller, D.C. & Goodnight, K.F. (1989). Estimating relatedness using genetic markers. Evolution, 43, 258-275.
- Ritland, K. (1996). Estimators for pairwise relatedness and individual inbreeding coefficients. Genetical Research, 67, 175-185.
- Wang, J. (2002). An estimator for pairwise relatedness using molecular markers. Genetics, 160, 1203-1215.
- Wang, J. (2007). Triadic IBD coefficients and applications to estimating pairwise relatedness. Genetical Research, 89, 135-153.

Examples

```
## Not run:
# download the program once per session (see Details)
dartRverse::gl.download.binary(software = "coancestry", out.dir = tempdir())
gl  <- dartR.data::platypus.gl[1:40, 1:1000]
res <- gl.relatedness(gl, estimators = c("wang", "lynchrd"),
                      coancestry.path = file.path(tempdir(), "coancestry"))
res$wang[1:5, 1:5]
head(res$dyads)

## End(Not run)
```

---

gl.report.errorrate *Estimates the genotyping error rate from replicated individuals*

Description

Estimates the per-allele genotyping error rate (epsilon) of SNP data from pairs of replicate samples of the same individual (technical replicates, re-genotyped samples, clones). The estimate is made for every replicate pair and for all pairs pooled, and can be used directly as the epsilon of gl.test.offspring.

Usage

```
gl.report.errorrate(
  x,
  pairs = NULL,
  replicates = NULL,
  min.loci = 100,
  alpha = 0.05,
  plot.display = TRUE,
  plot.theme = theme_dartR(),
  plot.colors = c("#2171B5", "#D7301F"),
  plot.file = NULL,
  plot.dir = NULL,
  verbose = NULL
)
```

Arguments

x	Name of the genlight object containing the SNP data [required].
pairs	A data frame with two columns giving the replicate pairs, as individual names or row indices of x. Use either pairs or replicates [default NULL].
replicates	A vector with one entry per individual in x giving its replicate group (e.g. the sample ID shared by all replicates of one individual). All pairs within each group are compared; individuals with NA, and groups with one member, are ignored [default NULL].

min.loci	Minimum number of SNPs genotyped in both samples of a pair for the pair to be used [default 100].
alpha	The confidence intervals have coverage 1 - alpha [default 0.05].
plot.display	If TRUE, the plot is displayed in the plot window [default TRUE].
plot.theme	Theme for the plot [default theme_dartR()].
plot.colors	Vector of two colours: per-pair estimates and pooled estimate [default c('#2171B5', '#D7301F')].
plot.file	Name for the RDS binary file to save (base name only, exclude extension) [default NULL].
plot.dir	Directory to save the plot RDS files [default as specified by the global working directory or tempdir()].
verbose	Verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log; 3, progress and results summary; 5, full report [default 2, unless specified using gl.set.verbosity].

Details

Model. Each allele copy is miscalled independently with probability epsilon, so a true homozygote is observed as a heterozygote with probability 2 epsilon (1 - epsilon) and a true heterozygote as either homozygote with probability epsilon (1 - epsilon). This is the error model of gl.test.offspring. Two replicates share the same true genotype, so the expected frequency of each observed genotype combination is P(g1, g2) = sum over t of f(t) P(g1 | t) P(g2 | t), where f(t) is the frequency of true genotype t across SNPs. Epsilon and f are estimated jointly by maximum likelihood from the 3 x 3 table of observed genotype combinations. f is estimated from the replicates themselves, not from the whole data set, because observed genotype frequencies include errors and the replicated individuals may come from a different population.

Estimates. For every pair, epsilon is estimated from that pair's table, with a profile-likelihood confidence interval. The pooled estimate uses the summed table of all pairs (pairs with more SNPs get more weight). The mean and median of the per-pair estimates are also reported; a mean clearly above the pooled estimate, or pairs far outside the pooled interval, point to a few bad samples. When a group has three or more replicates, its pairs share samples and are not independent, so the pooled interval is too narrow.

Model fit. The fit of the pooled table is tested with a G-test (2 df). A poor fit has two usual causes: errors that are not symmetric per allele, typically allelic dropout, which turns heterozygotes into homozygotes much more often than the reverse (compare the observed and expected tables); or pairs with clearly different error rates (check the per-pair estimates and the plot). With many SNPs the test also picks up small departures that do not matter in practice.

Missing data. Loci called in one replicate but missing in the other are not errors in this model but are reported per pair as missing_discordance (proportion of loci called in at least one sample).

Value

Returns invisibly a list with the components:

- per.pair: data frame with one row per pair: ind1, ind2, group, loci (SNPs called in both), discordant (SNPs with different genotypes), discordance_rate, missing_discordance, epsilon, lower, upper.

- pooled: data frame with the pooled epsilon and its confidence interval, the mean and median of the per-pair estimates, and the number of pairs and SNP comparisons.
- genotype.freq: estimated true genotype frequencies (0, 1, 2).
- observed: pooled 3 x 3 table of observed genotype combinations (made symmetric, so each pair counts in both orders, halved).
- expected: expected table under the pooled estimate.
- fit: G statistic, df and p-value of the model fit.
- plot: the ggplot.

Author(s)

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

See Also

gl.test.offspring

Examples

```
# treat the first two individuals as replicates
res <- gl.report.errorrate(bandicoot.gl, pairs = data.frame(1, 2))
# replicate groups from a sample ID in the individual names, e.g.
# "AB123_rep1", "AB123_rep2"
# reps <- sub("_rep[0-9]+$", "", indNames(gl))
# res <- gl.report.errorrate(gl, replicates = reps)
```

---

gl.report.parent.offspring

Identifies putative parent offspring within a population

Description

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

Usage

```
gl.report.parent.offspring(
  x,
  min.rdepth = 12,
  min.reproducibility = 1,
  range = 1.5,
  plot.filters = FALSE,
  plot_theme = theme_dartR(),
  plot_colors = NULL,
  plot.dir = NULL,
  plot.file = NULL,
  verbose = NULL
)
```

Arguments

x	Name of the genlight object containing the SNP genotypes [required].
min.rdepth	Minimum read depth to include in analysis [default 12].
min.reproducibility	Minimum reproducibility to include in analysis [default 1].
range	Specifies the range to extend beyond the interquartile range for delimiting outliers [default 1.5 interquartile ranges].
plot.filters	Whether to show the plots of filters within the function [default FALSE].
plot_theme	Theme for the plot. See Details for options [default theme_dartR()].
plot_colors	List of two color names for the borders and fill of the plots [default NULL, which uses gl.colors(2)].
plot.dir	Directory to save the plot RDS files [default as specified by the global working directory or tempdir()]
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, progress log; 3, progress and results summary; 5, full report [default 2, unless specified using gl.set.verbosity].

Details

If two individuals are in a parent offspring relationship, the true number of pedigree inconsistent loci should be zero, but SNP calling is not infallible. Some loci will be miscalled. The problem thus becomes one of determining if the two focal individuals have a proportion of pedigree inconsistent loci lower than would be expected of typical unrelated individuals. The proportion is taken over the loci genotyped in both individuals, so individuals with much missing data are not flagged merely because fewer loci could be compared. Pairs are flagged when their proportion lies strictly below the first quartile minus range times the interquartile range of all pairs. There are some quite sophisticated software packages available to formally apply likelihoods to the decision, but we use a simple outlier comparison.

To reduce the frequency of miss-calls, and so emphasize the difference between true parent-offspring pairs and unrelated pairs, the data can be filtered on read depth.

Typically minimum read depth is set to 5x, but you can examine the distribution of read depths with the function gl.report.rdepth and push this up with an acceptable loss of loci. 12x might be a good minimum for this particular analysis. It is sensible also to push the minimum reproducibility up to 1, if that does not result in an unacceptable loss of loci. Reproducibility is stored in the slot @other$loc.metrics$RepAvg and is defined as the proportion of technical replicate assay pairs for which the marker score is consistent. You can examine the distribution of reproducibility with the function gl.report.reproducibility.

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

The function gl.filter.parent.offspring will filter out those individuals in a parent offspring relationship.

Note that if your dataset does not contain RepAvg or rdepth among the locus metrics, the filters for reproducibility and read depth are not used. 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 data frame of putative parent-offspring pairs, ordered by prop, with columns: Outlier, the number of pedigree inconsistent loci; ind1 and ind2, the two individuals; n.loci, the number of loci genotyped in both; prop, Outlier / n.loci; zscore, the standardised prop; and p, the one-sided (lower tail) normal probability of zscore. An empty data frame with the same columns if no parent-offspring relationships were found.

Author(s)

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

See Also

gl.report.rdepth ,gl.report.reproducibility, gl.filter.parent.offspring

Examples

```
if (isTRUE(getOption("dartR_fbm"))) testset.gl <- gl.gen2fbm(testset.gl)
out <- gl.report.parent.offspring(testset.gl[1:10, 1:100])
```

gl.run.colony *Run COLONY2*

Description

An R wrapper for the COLONY pedigree-inference software (Jones & Wang 2010), to run full-pedigree likelihood analyses of SNP genotypes from R. The function writes the COLONY input file with gl2colony, runs the COLONY executable and reads the best configuration back into R.

Usage

```
gl.run.colony(
  x,
  colony.path = getwd(),
  outfile = "colony2.dat",
  outpath = NULL,
  project.name = "my_project",
  output.name = "my_project",
  probability.father = 0.5,
  probability.mother = 0.5,
  seed = NULL,
  update.allele.freq = 0,
  di.mono.ecious = 2,
  inbreed = 0,
  haplodiploid = 0,
  polygamy.male = 0,
  polygamy.female = 0,
  clone.inference = 1,
  scale.shibship = 1,
  sibship.prior = 0,
  known.allele.freq = 0,
  num.runs = 1,
  length.run = 2,
  monitor.method = 0,
  monitor.interval = 10000,
  windows.gui = 0,
  likelihood = 0,
  precision.fl = 2,
  marker.id = "mk@",
  marker.type = "0@",
  allelic.dropout = "0.000@",
  other.typ.err = "0.05@",
  paternity.exclusion.threshold = "0 0",
  maternity.exclusion.threshold = "0 0",
  paternal.sibship = 0,
  maternal.sibship = 0,
  excluded.paternity = 0,
```

```
  excluded.maternity = 0,
  excluded.paternal.sibships = 0,
  excluded.maternity.sibships = 0,
  verbose = NULL
)
```

Arguments

x	A genlight object with SNP data and individual metadata columns 'offspring', 'mother', and 'father' indicating 'yes'/'no' for each sample. Column names and values are matched ignoring case. A missing column is filled with 'yes' (offspring) or 'no' (mother, father). Individual names must not contain whitespace [required].
colony.path	Path to the folder that contains the COLONY executable: Colony2p.exe (Windows), colony2s.ifort.out (Linux) or colony2s.out (macOS) [default getwd()].
outfile	File name of the output file (including extension) [default "colony2.dat"].
outpath	Folder for the COLONY input file and for all the output files COLONY writes [default global working directory or if not specified, tempdir()].
project.name	Project name to include in the file header [default 'my_project'].
output.name	Output name to include in the file header [default 'my_project'].
probability.father	Probability that the father of an offspring is included among candidates [default 0.5].
probability.mother	Probability that the mother of an offspring is included among candidates [default 0.5].
seed	Seed for the random number generator [default NULL].
update.allele.freq	0 = do not update allele frequencies; 1 = update [default 0].
di.mono.ecious	2 = dioecious species; 1 = monoecious species [default 2].
inbreed	0 = no inbreeding; 1 = inbreeding allowed [default 0].
haplodiploid	0 = diploid species; 1 = haplodiploid species [default 0].
polygamy.male	0 = polygamy; 1 = monogamy for males [default 0].
polygamy.female	0 = polygamy; 1 = monogamy for females [default 0].
clone.inference	0 = no clone inference; 1 = infer clones [default 1].
scale.shibship	0 = do not scale full sibship; 1 = scale [default 1].
sibship.prior	Sibship prior; only 0 (no prior) is supported, because other values require mean sibship sizes that this function does not write [default 0].
known.allele.freq	0 = unknown allele frequencies. Known frequencies (1) are not supported [default 0].
num.runs	Number of runs [default 1].

`length.run` 1-4 specifying run length (short, medium, long, very long) [default 2].

`monitor.method` 0 = monitor by iteration number; 1 = monitor by time (seconds) [default 0].

`monitor.interval`
Interval for monitoring (either iteration count or seconds) [default 10000].

`windows.gui` 0 = no Windows GUI; 1 = use Windows GUI [default 0].

`likelihood` 0-2 specifying likelihood scoring (PairLikelihood, FullLikelihood, FPLS) [default 0].

`precision.fl` 0-3 specifying precision level for full-likelihood (Low, Medium, High, VeryHigh) [default 2].

`marker.id` Marker IDs string; a trailing '@' applies one value to all loci [default 'mk@'].

`marker.type` Marker types string (0@ for codominant) [default '0@'].

`allelic.dropout`
Allelic dropout rate string; a single value without '@' (e.g. '0.01') is applied to all loci [default '0.000@'].

`other.typ.err` Other typing error rate string; a single value without '@' is applied to all loci [default '0.05@'].

`paternity.exclusion.threshold`
Number of offspring with known father and the exclusion threshold. Only a count of 0 is supported, because other values require a list of offspring-father pairs [default '0 0'].

`maternity.exclusion.threshold`
Number of offspring with known mother and the exclusion threshold. Only a count of 0 is supported [default '0 0'].

`paternal.sibship`
Number of known paternal sibships; only 0 is supported [default 0].

`maternal.sibship`
Number of known maternal sibships; only 0 is supported [default 0].

`excluded.paternity`
Number of offspring with excluded paternity; only 0 is supported [default 0].

`excluded.maternity`
Number of offspring with excluded maternity; only 0 is supported [default 0].

`excluded.paternal.sibships`
Number of excluded paternal sibships; only 0 is supported [default 0].

`excluded.maternity.sibships`
Number of excluded maternal sibships; only 0 is supported [default 0].

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

Details

COLONY implements a full-pedigree likelihood method that simultaneously infers sibships and parentage by considering the likelihood of entire pedigree configurations rather than pairwise comparisons.

COLONY writes its output files (named after output.name) into outpath.

COLONY truncates individual IDs to 20 characters and splits them at spaces. If any name is longer than 20 characters or contains whitespace, COLONY runs on the IDs ind1, ind2, ... instead; the original names are restored in best.config, and the file output.name.IDmap.csv in outpath maps the IDs used in COLONY's own output files to the original names. COLONY does not return an error status when it rejects its input, so the function checks for the COLONY error file and for a new BestConfig file and stops with COLONY's message if the run failed.

Value

Invisibly, a list with two elements:

- files – full paths of the output files written by this run (including output.name.IDmap.csv when IDs were replaced).
- best.config – the best configuration (output.name.BestConfig) as a data frame with columns OffspringID, FatherID, MotherID, CloneIndex and ClusterIndex. Parents not among the candidates are given COLONY's inferred labels (e.g. *1 for fathers, #1 for mothers).

Author(s)

Author(s): Jesús Castrejón-Figueroa, Diana A. Robledo-Ruiz, Luis Mijangos. Custodian: Luis Mijangos – Post to https://groups.google.com/d/forum/dartr

References

Jones, O. R., & Wang, J. (2010). COLONY: a program for parentage and sibship inference from multilocus genotype data. Molecular Ecology Resources, 10(3), 551-555.

See Also

Other captive management: gl.diagnostics.relatedness(), gl.parentage.power(), gl.plot.network(), gl.run.EMIBD9(), gl.sim.relatedness(), gl2colony()

Examples

```
## Not run:
# requires the COLONY executable
t1 <- testset.gl[1:30, 1:50]
t1@other$ind.metrics$offspring <- rep(c("yes", "no", "no"), each = 10)
t1@other$ind.metrics$father <- rep(c("no", "yes", "no"), each = 10)
t1@other$ind.metrics$mother <- rep(c("no", "no", "yes"), each = 10)
res <- gl.run.colony(t1, colony.path = "path/to/colony", seed = 1234)
head(res$best.config)

## End(Not run)
```

gl.run.EMIBD9 *Run program EMIBD9*

Description

Runs the program EMIBD9 (Wang 2022) on a genlight object with SNP data and returns pairwise kinship, the IBD mode probabilities (delta1 to delta9) and individual inbreeding.

Usage

```
gl.run.EMIBD9(
  x,
  outfile = "EMIBD9_Res.ibd9",
  outpath = tempdir(),
  emibd9.path = getwd(),
  OutAlleleFre = 0,
  EM_Method = 1,
  Inbreed = FALSE,
  palette_convergent = NULL,
  parallel = FALSE,
  ncores = 1,
  ISeed = 42,
  plot.out = TRUE,
  plot.dir = NULL,
  plot.file = NULL,
  verbose = NULL
)
```

Arguments

x	Name of the genlight object containing the SNP data [required].
outfile	File name (without a path) of the EMIBD9 output file; it is copied to outpath [default "EMIBD9_Res.ibd9"].
outpath	Path where to save the output file. Use outpath=getwd() or outpath='.' when calling this function to direct output files to your working or current directory [default tempdir(), mandated by CRAN].
emibd9.path	Path to the folder with the EMIBD9 files. Please note there are 2 different executables depending on your OS: EM_IBD_P.exe (=Windows) EM_IBD_P (=Mac, Linux). You only need to point to the folder (the function will recognise which OS you are running) [default getwd()].
OutAlleleFre	Whether to output allele frequencies (TRUE/FALSE or 1/0) [default FALSE].
EM_Method	What the expectation maximization (EM) algorithm estimates: 0, the IBD coefficients only (allele frequencies are not updated); 1, the IBD coefficients and allele frequencies jointly [default 1].
Inbreed	A boolean that indicates whether to compute inbreeding (i.e. delta1 to delta6) [default FALSE].

palette_convergent
: A character vector of colours to use for the heatmap plot. If NULL, the default palette from gl.colors("div") will be used [default NULL].

parallel
: A boolean that indicates whether to run the parallel version of EM IBD9 (EM_IBD_P_mpi) [default FALSE].

ncores
: An integer specifying the number of cores to use when parallel is TRUE [default 1].

ISeed
: An integer specifying the random seed to use for the EM algorithm [default 42].

plot.out
: A boolean that indicates whether to plot the results [default TRUE].

plot.dir
: Directory to save the plot RDS files [default as specified by the global working directory or tempdir()]

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, progress log, including the EMIBD9 console output; 3, progress and results summary; 5, full report [default 2, unless specified using gl.set.verbosity].

Details

The results of EMIBD9 include the identical in state (IIS) values for each mode (S1 - 9) and nine condensed identical by descent (IBD) modes (delta1 - delta9) as well as the relatedness coefficient (r). Alleles are IIS if they are the same. Similarly, IBD describes a matching allele between two individuals that has been inherited from a common ancestor or common gene. In a pairwise comparison, delta1 to delta9 are the probabilities associated with each IBD mode. delta1 to delta6 take value > 0 in presence of inbreeding and hence are only computed when this option is selected.

EMIBD9 uses an expectation maximization (EM) algorithm based on the maximum likelihood expectations (MLE) of δ to estimate both allele frequencies (p) and δ jointly from genotype data. By iteratively calculating p and δ, relatedness can be modified to reduce biases due to small sample sizes. Wang J. (2022) suggest the resulting r coefficient is therefore more robust compared to previous methods.

EM_Method = 0 keeps the allele frequencies fixed at their sample values; only EM_Method = 1 updates them and uses random starting values for δ.

The kinship coefficient is the probability that two alleles at a random locus drawn from two individuals are IBD.

Below is a table modified from Speed & Balding (2015) showing kinship values, and their confidence intervals (CI), for different relationships.

Relationship	**Kinship**	**95% CI**
Identical twins / clones / same individual	0.5	-
Sibling / Parent-Offspring	0.25	(0.204, 0.296)
Half-sibling	0.125	(0.092, 0.158)
First cousin	0.062	(0.038, 0.089)
Half-cousin	0.031	(0.012, 0.055)
Second cousin	0.016	(0.004, 0.031)
Half-second cousin	0.008	(0.001, 0.020)

Third cousin	0.004	(0.000, 0.012)
Unrelated	0	-

For greater detail on the methods employed by EMIBD9, we encourage you to read Wang, J. (2022).

Download the program from here:

https://www.zsl.org/about-zsl/resources/software/emibd9

For Windows, Mac and Linux install the program then point to the folder where you find: EM_IBD_P.exe (=Windows) and EM_IBD_P (=Mac, Linux). If running really slow you may want to create the files using the function and then run in parallel using the documentation provided by the authors [you need to have mpiexec installed].

Please note individual names must have a maximal length of 20 characters. The IDs must NOT contain blank space and other illegal characters (such as /), and must be unique among all sampled individuals (i.e. NO duplications). Any string longer than 20 characters for individual ID will be truncated to have 20 characters.

Each call runs EMIBD9 in its own temporary folder. EMIBD9 does not return an error status when it fails, so the function stops with the last lines of the EMIBD9 console output when no output file is written.

Value

A list with four elements:

- rel – a square matrix of pairwise kinship coefficients (theta; the EMIBD9 column r(1,2)), with self-comparisons on the diagonal (0.5 x (1 + F)), tagged attr(rel, "scale") = "kinship".
- raw – the raw EMIBD9 table, all pairs including self-comparisons, with numeric columns.
- processed – the table without self-comparisons and redundant pairs (e.g. the second individual with the first, when the first with the second is already present).
- inbreeding – a table of individual inbreeding values; EMIBD9 writes it whether or not Inbreed is TRUE.

Author(s)

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

References

- Wang, J. (2022). A joint likelihood estimator of relatedness and allele frequencies from a small sample of individuals. Methods in Ecology and Evolution, 13(11), 2443-2462.

See Also

Other captive management: gl.diagnostics.relatedness(), gl.parentage.power(), gl.plot.network(), gl.run.colony(), gl.sim.relatedness(), gl2colony()

Examples

```
## Not run:
#To run this function needs EMIBD9 installed in your computer
if (isTRUE(getOption("dartR_fbm"))) platypus.gl <- gl.gen2fbm(platypus.gl)
t1 <- gl.filter.allna(platypus.gl)
res_rel <- gl.run.EMIBD9(t1)

## End(Not run)
```

`gl.sim.relatedness` *Simulate kinship estimates for known relationships*

Description

Simulates pairs of individuals with a known relationship (full siblings, half siblings or first cousins) from the individuals in a genlight object, estimates their kinship with EMIBD9 (Wang, J. 2022) and summarises the distribution of the estimates over replicates. The result shows which kinship values that relationship produces in this dataset, to guide the choice of kinship thresholds.

Usage

```
gl.sim.relatedness(
  x,
  rel = "full.sib",
  nboots = 10,
  emibd9.path = getwd(),
  conf = 0.95,
  OutAlleleFre = 0,
  EM_Method = 1,
  Inbreed = FALSE,
  ISeed = 42,
  parallel = FALSE,
  ncores = 1,
  plot.out = TRUE,
  plot.dir = NULL,
  plot.file = NULL,
  verbose = NULL
)
```

Arguments

`x`	Name of the genlight object containing the SNP data [required].
`rel`	The relationship to simulate. One of 'full.sib', 'half.sib', 'first.cousin' [default 'full.sib'].
`nboots`	The number of simulated pairs (replicates); use at least 100 for a stable interval [default 10].

emibd9.path	The location of all necessary files to run EMIBD9 (read more at gl.run.EMIBD9) [required].
conf	The proportion of simulated kinship values contained in the reported interval, and the confidence level of the interval for the mean [default 0.95].
OutAlleleFre	Whether to write (1) or not (0) the EMIBD9 allele frequency file [default 0].
EM_Method	What the EMIBD9 expectation maximization estimates: 0, the IBD coefficients only (allele frequencies are not updated); 1, the IBD coefficients and allele frequencies jointly (see gl.run.EMIBD9) [default 1].
Inbreed	Whether EMIBD9 allows inbreeding when estimating IBD coefficients [default FALSE].
ISeed	Seed for the EMIBD9 random number generator. The simulation of offspring uses R's random number generator; set it with set.seed() for reproducible results [default 42].
parallel	Use the parallel (MPI) version of EMIBD9. Only works for Mac and Linux at the moment [default FALSE].
ncores	How many cores should be used [default 1].
plot.out	A boolean that indicates whether to plot the results [default TRUE].
plot.dir	Directory to save the plot RDS files [default as specified by the global working directory or tempdir()]
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, progress log; 3, progress and results summary; 5, full report [default 2, unless specified using gl.set.verbosity].

Details

In each replicate, parents are drawn at random, without replacement, from the individuals in x, and the relatives are simulated with `dartR.sim::gl.sim.offspring`:

- full.sib: two offspring of the same two parents.
- half.sib: one offspring of each of two mothers with the same father.
- first.cousin: two full siblings, each mated with a further (different) individual; the two resulting offspring are first cousins.

The simulated individuals are added to x and kinship is estimated with `gl.run.EMIBD9`, so allele frequencies come from the sample.

The interval reported is the range that contains the central conf proportion of the simulated kinship values (e.g. the 2.5% and 97.5% quantiles for conf = 0.95). It describes the spread of individual pairs and is the one to compare with thresholds. The confidence interval of the mean is reported separately; it narrows as nboots grows.

Below is a table modified from Speed & Balding (2015) showing kinship values, and their confidence intervals (CI), for different relationships.

Relationship	Kinship	95% CI

Identical twins / clones / same individual	0.5	-
Sibling / Parent-Offspring	0.25	(0.204, 0.296)
Half-sibling	0.125	(0.092, 0.158)
First cousin	0.062	(0.038, 0.089)
Half-cousin	0.031	(0.012, 0.055)
Second cousin	0.016	(0.004, 0.031)
Half-second cousin	0.008	(0.001, 0.020)
Third cousin	0.004	(0.000, 0.012)
Unrelated	0	-

Value

Invisibly, a named list:

- values – the simulated kinship values, one per replicate.
- mean – their mean.
- interval – the lower and upper limits containing the central conf proportion of the values.
- ci.mean – the confidence interval (level conf) of the mean.
- plot – a histogram of the values with the mean and the interval.

Author(s)

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

References

- Wang, J. (2022). A joint likelihood estimator of relatedness and allele frequencies from a small sample of individuals. Methods in Ecology and Evolution, 13(11), 2443-2462.
- Speed, D., Balding, D. (2015). Relatedness in the post-genomic era: is it still useful? Nature Reviews Genetics 16, 33-44.

See Also

Other captive management: gl.diagnostics.relatedness(), gl.parentage.power(), gl.plot.network(), gl.run.EMIBD9(), gl.run.colony(), gl2colony()

Examples

```
## Not run:
# To run this function needs EMIBD9 installed in your computer
t1 <- gl.filter.allna(platypus.gl)
set.seed(1)
res <- gl.sim.relatedness(t1, rel = "half.sib", nboots = 100,
                          emibd9.path = "path/to/emibd9")
res$interval

## End(Not run)
```

`gl.test.offspring` *Tests candidate parent pairs for offspring using Mendelian mismatches and LOD scores*

Description

Tests whether two candidate parents can be the parents of an offspring from biallelic SNP data. For each trio (parent 1, parent 2, offspring) the number of SNPs whose genotypes are Mendelian-incompatible is compared with the number expected from genotyping error alone, and a LOD score ranks the candidate parent pairs of each offspring.

Usage

```
gl.test.offspring(
  x = NULL,
  p1 = NULL,
  p2 = NULL,
  off = NULL,
  epsilon = 0.01,
  allele.freq = NULL,
  alpha = 0.05,
  min.loci = 100L,
  return.loci = TRUE,
  max.tests = 100000L,
  plot.display = TRUE,
  plot.theme = NULL,
  plot.colors = c("#2171B5", "#D7301F"),
  plot.file = NULL,
  plot.dir = tempdir(),
  plot.top = 3L,
  plot.max = 30L,
  verbose = 2L
)
```

Arguments

`x`	Genlight object for exploratory mode: every individual is tested as offspring of every unordered pair of the other individuals. Use either x or p1, p2 and off [default NULL].
`p1`	Genlight object of candidate first parents (e.g. fathers) [default NULL].
`p2`	Genlight object of candidate second parents (e.g. mothers) [default NULL].
`off`	Genlight object of offspring to test [default NULL].
`epsilon`	Per-allele genotyping error probability, applied equally to parents and offspring; must be in [0, 0.5) [default 0.01].

allele.freq	Allele frequencies (of the allele counted by the genotype 0/1/2 coding): a single value for all SNPs, a vector with one value per SNP in locus order, or a vector named by locus. If NULL, they are estimated from all unique individuals in the inputs [default NULL].
alpha	Significance level of the mismatch test and coverage of the prediction interval (1 - alpha) [default 0.05].
min.loci	Minimum number of SNPs genotyped in all three individuals for a trio to be tested; trios with fewer SNPs get NA [default 100].
return.loci	If TRUE, the names of the mismatching SNPs of each trio are returned [default TRUE].
max.tests	Maximum number of trios to test; the function stops if more would be needed [default 100000].
plot.display	If TRUE, the plots are displayed in the plot window [default TRUE].
plot.theme	Theme for the plots [default theme_dartR()].
plot.colors	Vector of two colours. Checked but currently not used by the plots [default c('#2171B5', '#D7301F')].
plot.file	Name of the file to save the plots to with ggsave(), e.g. "parentage.png" (".png" is added if no extension is given). The heatmap is saved with the suffix "_heatmap" [default NULL].
plot.dir	Directory to save the plot files to [default tempdir()].
plot.top	Number of highest-LOD trios per offspring shown in the candidate plot [default 3].
plot.max	Maximum number of trios shown in the candidate plot [default 30].
verbose	Verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log; 3, progress and results summary; 5, full report [default 2].

Details

Modes. In candidate mode (p1, p2 and off given) every combination of a parent from p1, a parent from p2 and an offspring from off is tested, excluding combinations in which the same individual (by name) appears twice. If an individual is in both p1 and p2 (e.g. when sexes are unknown and p1 = p2), each unordered parent pair is tested once per offspring, because swapping the two parents gives identical results; such individuals must have identical genotypes in p1 and p2. In exploratory mode (x given) each individual is tested as offspring of each unordered pair of the remaining individuals, i.e. n(n - 1)(n - 2)/2 trios, which grows quickly with n. The three objects must contain the same SNPs; if all have locus names, loci are matched by name, otherwise by position.

Mismatch test. A SNP is Mendelian-incompatible if the observed offspring genotype cannot arise from the observed parental genotypes (e.g. both parents 0 and offspring 1). True parents can still show mismatches because of genotyping error. Errors are modelled per allele with probability epsilon, independently in parents and offspring. Given the observed parental genotypes, Hardy-Weinberg priors and the error model, the probability that each SNP shows a mismatch under the null hypothesis "these are the parents" is computed. The number of mismatches then follows a Poisson-binomial distribution, which gives the expected number of mismatches, an exact (1 - alpha)

prediction interval and the p-value P(mismatches >= observed). Trios with p-value < alpha are excluded (reject_H0 = TRUE).

The Poisson-binomial is evaluated only over the range of counts that carries probability (the tail beyond it is < 1e-15), with one FFT per trio, and trios are processed in vectorised batches.

LOD. The LOD score is the log10 ratio of the likelihood of the offspring genotypes given both observed parents to their likelihood as an unrelated individual drawn from the population, both under the same error model. Positive values favour parentage. delta_LOD is the difference to the highest LOD of the same offspring (0 for the best trio). LOD ranks candidates of an offspring; it is not a universal assignment threshold.

Allele frequencies. If allele.freq is NULL, frequencies are estimated from the unique individuals in the inputs, with Beta(1,1) shrinkage and a correction for genotyping error. With few individuals, or when the inputs are a small family group, these estimates are imprecise and affect both the expected mismatches and LOD; supplying frequencies estimated from a larger population sample is recommended.

Plots. Two ggplots are produced. The candidate plot shows, for the top plot.top LOD trios of each offspring (plus all non-excluded trios, up to plot.max), the observed mismatches (coloured by LOD, shape by test result), the expected mismatches (white point) and the prediction interval (grey bar). The heatmap shows observed mismatches for all evaluable trios, offspring by parental pair, with non-excluded trios circled and the highest-LOD trio of each offspring starred.

Value

A list with the components:

- results: data frame with one row per trio: test_id, offspring, parent1, parent2, loci (SNPs genotyped in all three), observed_mismatches, expected_mismatches, prediction_lower, prediction_upper, expected_mismatch_rate, p_value, LOD, reject_H0 and delta_LOD.
- mismatch.loci: list with the mismatching SNP names of each trio (NULL if return.loci = FALSE).
- plot: the candidate ggplot (NULL if no trio could be tested).
- heatmap: the heatmap ggplot (NULL if no trio has a finite LOD).
- heatmap.info: list with the numbers of trios in total, with a finite LOD, and plotted in the heatmap.
- allele.freq: named vector of the allele frequencies used.
- settings: list with epsilon, alpha, the source of the allele frequencies, the mode, min.loci and the number of trios tested.
- plot.info: list with the numbers of trios tested and shown in the candidate plot.

Author(s)

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

See Also

gl.report.parent.offspring, gl.filter.parent.offspring

Examples

```
# three candidate fathers and three candidate mothers
x <- platypus.gl[1:6, 1:500]
dads <- x[1:3, ]
mums <- x[4:6, ]
# offspring of the first father with the first two mothers
kids <- gl.sim.offspring(fathers = dads[1, ], mothers = mums[1:2, ],
                         noffpermother = 2, verbose = 0)
res <- gl.test.offspring(p1 = dads, p2 = mums, off = kids, verbose = 0)
# best candidate pairs first
head(res$results[order(-res$results$LOD), ])


# exploratory mode: all individuals against all pairs of the others
res2 <- gl.test.offspring(x = rbind(x[c(1, 4, 5)], kids[1:2]), verbose = 0)
```

`gl2colony` *Export a COLONY2 input file from a genlight object*

Description

Export a formatted text file compatible with the COLONY2 software from a `genlight` object containing parental and offspring information stored in the individual metadata.

Usage

```
gl2colony(
  x,
  outfile = "colony2.dat",
  outpath = NULL,
  project.name = "my_project",
  output.name = "my_project",
  probability.father = 0.5,
  probability.mother = 0.5,
  seed = NULL,
  update.allele.freq = 0,
  di.mono.ecious = 2,
  inbreed = 0,
  haplodiploid = 0,
  polygamy.male = 0,
  polygamy.female = 0,
  clone.inference = 1,
  scale.shibship = 1,
  sibship.prior = 0,
  known.allele.freq = 0,
  num.runs = 1,
```

```
  length.run = 2,
  monitor.method = 0,
  monitor.interval = 10000,
  windows.gui = 0,
  likelihood = 0,
  precision.fl = 2,
  marker.id = "mk@",
  marker.type = "0@",
  allelic.dropout = "0.000@",
  other.typ.err = "0.05@",
  paternity.exclusion.threshold = "0 0",
  maternity.exclusion.threshold = "0 0",
  paternal.sibship = 0,
  maternal.sibship = 0,
  excluded.paternity = 0,
  excluded.maternity = 0,
  excluded.paternal.sibships = 0,
  excluded.maternity.sibships = 0,
  verbose = NULL
)
```

Arguments

x	A genlight object with SNP data and individual metadata columns 'offspring', 'mother', and 'father' indicating 'yes'/'no' for each sample. Column names and values are matched ignoring case. A missing column is filled with 'yes' (offspring) or 'no' (mother, father). Individual names must not contain whitespace [required].
outfile	File name of the output file (including extension) [default "colony2.dat"].
outpath	Path where to save the output file [default global working directory or if not specified, tempdir()].
project.name	Project name to include in the file header [default 'my_project'].
output.name	Output name to include in the file header [default 'my_project'].
probability.father	Probability that the father of an offspring is included among candidates [default 0.5].
probability.mother	Probability that the mother of an offspring is included among candidates [default 0.5].
seed	Seed for the random number generator [default NULL].
update.allele.freq	0 = do not update allele frequencies; 1 = update [default 0].
di.mono.ecious	2 = dioecious species; 1 = monoecious species [default 2].
inbreed	0 = no inbreeding; 1 = inbreeding allowed [default 0].
haplodiploid	0 = diploid species; 1 = haplodiploid species [default 0].

`polygamy.male` 0 = polygamy; 1 = monogamy for males [default 0].

`polygamy.female`
0 = polygamy; 1 = monogamy for females [default 0].

`clone.inference`
0 = no clone inference; 1 = infer clones [default 1].

`scale.shibship` 0 = do not scale full sibship; 1 = scale [default 1].

`sibship.prior` Sibship prior; only 0 (no prior) is supported, because other values require mean sibship sizes that this function does not write [default 0].

`known.allele.freq`
0 = unknown allele frequencies. Known frequencies (1) are not supported [default 0].

`num.runs` Number of runs [default 1].

`length.run` 1-4 specifying run length (short, medium, long, very long) [default 2].

`monitor.method` 0 = monitor by iteration number; 1 = monitor by time (seconds) [default 0].

`monitor.interval`
Interval for monitoring (either iteration count or seconds) [default 10000].

`windows.gui` 0 = no Windows GUI; 1 = use Windows GUI [default 0].

`likelihood` 0-2 specifying likelihood scoring (PairLikelihood, FullLikelihood, FPLS) [default 0].

`precision.fl` 0-3 specifying precision level for full-likelihood (Low, Medium, High, VeryHigh) [default 2].

`marker.id` Marker IDs string; a trailing '@' applies one value to all loci [default 'mk@'].

`marker.type` Marker types string (0@ for codominant) [default '0@'].

`allelic.dropout`
Allelic dropout rate string; a single value without '@' (e.g. '0.01') is applied to all loci [default '0.000@'].

`other.typ.err` Other typing error rate string; a single value without '@' is applied to all loci [default '0.05@'].

`paternity.exclusion.threshold`
Number of offspring with known father and the exclusion threshold. Only a count of 0 is supported, because other values require a list of offspring-father pairs [default '0 0'].

`maternity.exclusion.threshold`
Number of offspring with known mother and the exclusion threshold. Only a count of 0 is supported [default '0 0'].

`paternal.sibship`
Number of known paternal sibships; only 0 is supported [default 0].

`maternal.sibship`
Number of known maternal sibships; only 0 is supported [default 0].

`excluded.paternity`
Number of offspring with excluded paternity; only 0 is supported [default 0].

`excluded.maternity`
Number of offspring with excluded maternity; only 0 is supported [default 0].

`excluded.paternal.sibships`
: Number of excluded paternal sibships; only 0 is supported [default 0].

`excluded.maternity.sibships`
: Number of excluded maternal sibships; only 0 is supported [default 0].

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

Details

This function formats and writes a COLONY2-compatible text file, including header, offspring genotypes, parental candidate probabilities, and candidate genotypes, based on the `genlight` object's individual metadata and genotype matrix.

Value

Invisibly returns the output filename.

Author(s)

Author(s): Jesús Castrejón-Figueroa, Diana A. Robledo-Ruiz. Custodian: Luis Mijangos – Post to https://groups.google.com/d/forum/dartr

References

Jones, O. R., & Wang, J. (2010). COLONY: a program for parentage and sibship inference from multilocus genotype data. Molecular Ecology Resources, 10(3), 551-555.

See Also

Other captive management: `gl.diagnostics.relatedness()`, `gl.parentage.power()`, `gl.plot.network()`, `gl.run.EMIBD9()`, `gl.run.colony()`, `gl.sim.relatedness()`

Examples

```
t1 <- testset.gl[1:30, 1:50]
t1@other$ind.metrics$offspring <- rep(c("yes", "no", "no"), each = 10)
t1@other$ind.metrics$father <- rep(c("no", "yes", "no"), each = 10)
t1@other$ind.metrics$mother <- rep(c("no", "no", "yes"), each = 10)
gl2colony(x = t1,
          outpath = tempdir(),
          seed = 1234,
          probability.father = 0.6,
          probability.mother = 0.4,
          allelic.dropout = '0.01',
          other.typ.err = '0.001')
```

Index

∗ **captive management**
gl.diagnostics.relatedness, 3
gl.parentage.power, 20
gl.plot.network, 23
gl.run.colony, 33
gl.run.EMIBD9, 37
gl.sim.relatedness, 40
gl2colony, 46
∗ **filter functions**
gl.filter.parent.offspring, 9
∗ **inbreeding functions**
gl.grm, 11
gl.grm.network, 13
gl.inbreeding, 17
∗ **relatedness functions**
gl.test.offspring, 43
∗ **relatedness**
gl.relatedness, 25
∗ **report functions**
gl.report.parent.offspring, 30
∗ **reporting functions**
gl.report.errorrate, 28

A.mat, *11, 12*

gl.assign.grm, 2
gl.diagnostics.relatedness, 3, *22*, *24*, *36*, *39*, *42*, *49*
gl.filter.callrate, *8*
gl.filter.parent.offspring, 9, *32*, *45*
gl.grm, 11, *16*, *19*
gl.grm.network, 13, *13*, *19*
gl.inbreeding, *13*, *16*, 17
gl.parentage.power, *8*, 20, *24*, *36*, *39*, *42*, *49*
gl.plot.network, *8*, *22*, 23, *36*, *39*, *42*, *49*
gl.relatedness, *8*, *19*, 25
gl.report.errorrate, 28
gl.report.parent.offspring, *9–11*, 30, *45*
gl.report.rdepth, *10, 11*, *32*
gl.report.reproducibility, *10, 11*, *32*
gl.run.colony, *8*, *22*, *24*, 33, *39*, *42*, *49*
gl.run.EMIBD9, *8*, *19*, *22*, *24*, *36*, 37, *41, 42*, *49*
gl.sim.relatedness, *8*, *22*, *24*, *36*, *39*, 40, *49*
gl.sim.WF.run, *8*
gl.test.offspring, *28*, *30*, 43
gl2colony, *8*, *22*, *24*, *36*, *39*, *42*, 46

layout_with_fr, *15*
layout_with_graphopt, *15*
layout_with_kk, *15*
layout_with_mds, *15*