Package: coralesce (via r-universe)

July 10, 2026

Title Group Coral Colonies Based on Genetic Relatedness

Version 1.0.1

Description Assigns coral colonies to clonal groups (genets) from single nucleotide polymorphism (SNP) data by comparing alleles at each locus across all pairwise combinations of colonies, and computes individual- and population-level mean kinship and gene diversity (excluding invariant loci). Wrapper functions read genotype CSV files from a 'Data' folder and write results to a 'Results' folder.

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

Depends R (>= 3.5.0)

Imports dplyr, igraph, magrittr, stringi, stringr, tidyr, utils

Suggests knitr, quarto, rmarkdown, testthat (>= 3.0.0), R.rsp

Config/testthat/edition 3

LazyData true

VignetteBuilder knitr, quarto, R.rsp

Config/pak/sysreqs libglpk-dev libicu-dev libxml2-dev

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

Date/Publication 2026-07-10 16:11:43 UTC

RemoteUrl https://github.com/colinpshea/coralesce

RemoteRef HEAD

RemoteSha f1dbeb33520e4137f49fdaac9653b5056e1585bb

Contents

`calcPercentNotNull` *Calculate percent not null*

Description

Calculates the percentage of values in `x` that are not `NA` (i.e., that carry valid SNP data). Used to summarise the proportion of loci with usable data for a colony or a pairwise comparison.

Usage

```
calcPercentNotNull(x)
```

Arguments

`x`	A vector, matrix, or data frame.

Value

A single numeric value: the percentage of non-`NA` values in `x`.

checkforAllowableData *Flag values that are valid IUPAC allele pairs*

Description

Tests, element by element, whether the supplied values are allowable allele pairs as listed in the IUPAC reference table. Used to identify which columns of an input data set contain valid SNP data.

Usage

```
checkforAllowableData(x)
```

Arguments

x	A vector of values to test (typically one column of the input data).

Value

A logical vector the same length as x, TRUE where the value is a valid allele pair in IUPAC$Allelepairs.

classifyAllelePairs *Classify allele pairs at a single locus*

Description

Builds all pairwise combinations of colonies at one locus and flags, for each pair, whether the two colonies share the same allele. By default comparisons of each colony with itself are included (needed for genet assignment); set include_self = FALSE to return only comparisons with other colonies (used for kinship). Operates on one locus (column) at a time.

Usage

```
classifyAllelePairs(dataset, locus, include_self = TRUE)
```

Arguments

dataset	A data frame with Coral_ID in column 1 and single-letter IUPAC allele data for one locus per remaining column.
locus	The column index of the locus to classify.
include_self	Logical; include self-comparisons (default TRUE).

Value

A data frame with columns coral1, coral2, allele1, allele2, locus, and match (TRUE/FALSE, or NA when either allele is missing).

classifyAllelePairsOthers *Classify allele pairs at a single locus (excluding self-comparisons)*

Description

Convenience wrapper around `classifyAllelePairs()` that returns only comparisons of each colony with other colonies (no self-comparisons). Used by the kinship pipeline, which must exclude within-colony comparisons.

Usage

```
classifyAllelePairsOthers(dataset, locus)
```

Arguments

`dataset`	A data frame with `Coral_ID` in column 1 and single-letter `IUPAC` allele data for one locus per remaining column.
`locus`	The column index of the locus to classify.

Value

A data frame with columns `coral1`, `coral2`, `allele1`, `allele2`, `locus`, and `match`; self-comparisons are omitted.

collapseToGenets *Collapse colonies to one representative per genet*

Description

Reduces a genotype data frame to a single representative colony per genet, using a set of genet assignments from `runGenets()`. This is a pre-filter for clone-corrected analyses: feeding the result to the kinship pipeline yields genet-level (clone-corrected) gene diversity rather than the eligible-pool diversity computed over all colonies (see `runKinship()`).

The function only filters rows; it does not alter genotype values, so it can be applied to either raw paired-allele data (e.g., from `readGeneticData()`) or single-letter coded data (from `convertBasePairstoCodes()`). It matches rows on `Coral_ID`, so identifiers may contain any characters.

Usage

```
collapseToGenets(
  dataset,
  genetAssignment,
  representative = c("most_data", "first"),
  drop_unassigned = FALSE
)
```

Arguments

dataset	A data frame with a Coral_ID column (plus any genotype columns). Rows are selected from this frame.
genetAssignment	A genet-assignment data frame with at least Coral_ID and genet columns (and, ideally, pctNull), as returned per file by runGenets().
representative	How to choose the representative colony within a genet: "most_data" (default) keeps the colony with the most scored loci (lowest pctNull); "first" keeps the first colony listed. "most_data" falls back to "first" if no pctNull column is present.
drop_unassigned	Logical. "Unassigned" colonies are those with no real genet – i.e. colonies with inadequate data (their self-comparison fell below the PctNotNull threshold) or all-NA colonies, carrying a missing or NA-suffixed genet label. Because they were never placed in a genet, they are not clonal duplicates of one another. The default (FALSE) retains each as its own unit, so no colonies are silently discarded; this suits eligible-pool questions (every colony is a real candidate). Set TRUE to drop them for a cleaner genet-level population estimate, where these low-data, unplaceable colonies would otherwise add noise. Note that TRUE removes rows, so the result will have fewer colonies than the input.

Value

A subset of dataset containing one row per genet plus (unless dropped) each unassigned colony. Column structure and order are unchanged.

See Also

runGenets() to produce genetAssignment; runKinship() and kinshipCalcsNoInvar() for the kinship pipeline this feeds.

Examples

```
## Not run:
# Clone-corrected (genet-level) kinship and gene diversity:
g   <- runGenets(PctMatchThreshold = 90, PctNotNullThreshold = 50)
ga  <- g$genetAssignments[[1]]
raw <- readGeneticData(file.path(getwd(), "Data", "myfile.csv"))[[1]]

reduced <- collapseToGenets(raw, ga)                # one colony per genet

coded   <- convertBasePairstoCodes(reduced)
withDat <- isolateAllNAColonies(coded)[[1]]
noInv   <- omitInvariantLoci(withDat)
k       <- kinshipCalcsNoInvar(noInv, subset = FALSE)
k$PopAvgMKGD                                         # clone-corrected diversity

## End(Not run)
```

computeGenets *Assign colonies to genets from a genotype data frame*

Description

Data-frame entry point to the genet-assignment pipeline: takes a genotype data frame directly (no Data/Results folders) and returns genet assignments. This is the in-memory counterpart to runGenets(), useful when the genotypes are already in R – in particular, its output is the genetAssignment that collapseToGenets() needs, so the whole clone-corrected workflow can run without files.

Internally it runs the same steps as runGenets(): translate paired alleles to IUPAC codes, set aside all-NA colonies, compare all colony pairs, and assign genets (groupByGenets()). Results are keyed by Coral_ID.

Usage

```
computeGenets(
  data,
  PctMatchThreshold = NULL,
  PctNotNullThreshold = NULL,
  speciesCode = NULL,
  getPairwiseAlleleMatches = FALSE
)
```

Arguments

data	A genotype data frame with a Coral_ID column and paired-allele SNP columns (e.g., "A:G"), as produced by readGeneticData(). A MatchMaker_Index column, if present, is carried through and used to order the output. Missing values may be NA or "?".
PctMatchThreshold	Minimum percent allele match across loci for two colonies to be called a clone. Required.
PctNotNullThreshold	Minimum percent of loci with valid data required to trust a comparison. Required.
speciesCode	Optional short code prefixed to each genet label (e.g., "ACER" gives ACER_00001). If NULL (default) genet labels are the zero-padded number alone (00001). Colonies without a genet (inadequate or all-NA data) carry a label ending in NA.
getPairwiseAlleleMatches	Logical; if TRUE, also return the full pairwise comparison table. Default FALSE.

Value

If getPairwiseAlleleMatches = FALSE, a genet-assignment data frame with columns Coral_ID, (MatchMaker_Index if supplied,) genet, pctNull, AdequateData. If TRUE, a list of that data frame plus the pairwise table.

See Also

runGenets() for the folder-based workflow; collapseToGenets() to reduce to one colony per genet; computeKinship() for the kinship pipeline.

Examples

```
## Not run:
raw <- readGeneticData(file.path(getwd(), "Data", "myfile.csv"))[[1]]
ga  <- computeGenets(raw, PctMatchThreshold = 90, PctNotNullThreshold = 50)

# fully folder-free clone-corrected diversity:
computeKinship(collapseToGenets(raw, ga))

## End(Not run)
```

---

computeKinship *Compute kinship and gene diversity from a genotype data frame*

Description

Data-frame entry point to the kinship pipeline: takes a genotype data frame directly (no Data/Results folders) and returns individual- and population-level mean kinship and gene diversity. This is the in-memory counterpart to runKinship(), useful when the genotypes are already in R – for example after collapsing to one colony per genet with collapseToGenets() for a clone-corrected estimate.

Internally it runs the same steps as runKinship(): translate paired alleles to IUPAC codes (convertBasePairstoCodes()), set aside all-NA colonies (isolateAllNAColonies()), optionally drop invariant loci, and compute kinship (kinshipCalcsNoInvar()). Results are keyed by Coral_ID.

Usage

```
computeKinship(data, subset = FALSE, targetN = NULL)
```

Arguments

data	A genotype data frame with a Coral_ID column and paired-allele SNP columns (e.g., "A:G"), as produced by readGeneticData() or collapseToGenets(). A MatchMaker_Index column, if present, is ignored. Missing values may be NA or "?".
subset	Logical; if TRUE, also compute the targetN-colony summary. Requires targetN. Default FALSE.
targetN	Number of least-related colonies to retain when subset = TRUE. Must be >= 2.

Value

A list with PopAvgMKGD, MK_init, and MK_final (see kinshipCalcsNoInvar()).

See Also

runKinship() for the folder-based workflow; collapseToGenets() to reduce to one colony per genet before computing a clone-corrected result.

Examples

```
## Not run:
raw <- readGeneticData(file.path(getwd(), "Data", "myfile.csv"))[[1]]

# eligible-pool kinship (all colonies, ramets included):
computeKinship(raw)

# clone-corrected kinship (one colony per genet):
ga <- runGenets(PctMatchThreshold = 90, PctNotNullThreshold = 50)$genetAssignments[[1]]
computeKinship(collapseToGenets(raw, ga))

## End(Not run)
```

---

convertBasePairstoCodes

Convert base-pair genotypes to single-letter IUPAC codes

Description

Keeps only Coral_ID and columns whose values are all valid allele pairs (per the IUPAC table), translates the paired alleles (e.g., G:G) to single-letter IUPAC codes (e.g., G), and returns the result in wide format (one column per locus). Missing markers ("?") are converted to NA.

Usage

```
convertBasePairstoCodes(initdata)
```

Arguments

initdata	A data frame with a Coral_ID column and one or more columns of paired-allele SNP data. Non-conforming columns (e.g., site name, an invalid base pair, or the bookkeeping MatchMaker_Index) are dropped here; they are reported separately by handleError_ProhibitedData().

Value

A wide data frame with Coral_ID and one single-letter-code column per valid locus.

determineAllAlleleMatches

Assess allele matches across all loci (self and other comparisons)

Description

Applies `classifyAllelePairs()` to every locus (`2:ncol(dataset)`) and stacks the results. Includes both self- and other-comparisons, as required for genet assignment via `groupByGenets()`.

Usage

```
determineAllAlleleMatches(dataset)
```

Arguments

`dataset`	A data frame with `Coral_ID` in column 1 and single-letter `IUPAC` allele data in the remaining columns.

Value

A data frame with columns `coral1`, `coral2`, `allele1`, `allele2`, `locus`, and `match`, covering all pairwise comparisons (self and other) at every locus.

determineAllAlleleMatchesOthers

Assess allele matches across all loci (other comparisons only)

Description

Applies `classifyAllelePairsOthers()` to every locus (`2:ncol(dataset)`) and stacks the results. Excludes self-comparisons, as required for kinship via `kinshipCalcsNoInvar()`.

Usage

```
determineAllAlleleMatchesOthers(dataset)
```

Arguments

`dataset`	A data frame with `Coral_ID` in column 1 and single-letter `IUPAC` allele data in the remaining columns.

Value

A data frame with columns `coral1`, `coral2`, `allele1`, `allele2`, `locus`, and `match` for all comparisons of colonies with other colonies.

find_dups *Stop if any colony has multiple rows of SNP data*

Description

Downstream reshaping requires one row per colony. This check stops with an informative error, listing the duplicated `Coral_ID`s, if any colony appears more than once.

Usage

```
find_dups(df)
```

Arguments

`df`	A data frame with a `Coral_ID` column.

Value

Invisibly `df` if all `Coral_ID`s are unique; otherwise an error.

groupByGenets *Assign colonies to genets from pairwise allele matches*

Description

For every pairwise comparison of colonies, computes the percent of loci that match and the percent of loci with scorable data, applies the user thresholds to decide which pairs are clones with adequate data, and assigns colonies to genets via `returnGenetIdentity()`. Colonies whose self-comparison falls below the data threshold are flagged `AdequateData = No` and carried through with `genet = NA`.

Usage

```
groupByGenets(
  CoralAlleleData,
  AlleleMatchResults,
  PctMatchThreshold = NULL,
  PctNotNullThreshold = NULL,
  getPairwiseAlleleMatches = FALSE
)
```

Arguments

`CoralAlleleData`
: Wide single-letter allele data (`Coral_ID` plus one column per locus); used to attach each colony's `pctNull`.

`AlleleMatchResults`
: Pairwise allele-match results from `determineAllAlleleMatches()`.

`PctMatchThreshold`
: Minimum percent allele match for a pair to be called a clone. Required.

`PctNotNullThreshold`
: Minimum percent of scorable loci required to trust a comparison. Required.

`getPairwiseAlleleMatches`
: Logical; if `TRUE`, also return the full pairwise comparison table. Default `FALSE`.

Value

A list with `genetAssignment` (columns `Coral_ID`, `genet`, `pctNull`, `AdequateData`) and `pairwiseAlleleMatches` (the pairwise table, or `NULL`).

`handleError_allZeros` *Identify colonies with no valid SNP data*

Description

Finds colonies that are `NA` at every locus. Such colonies cannot be assigned a genet from data and are handled separately: they are appended to the genet-assignment output with `genet` = `XXXX_NA`, `pctNull` = `100`, and `AdequateData` = `No`. Emits a message listing the affected colonies when any are found.

Usage

```
handleError_allZeros(dataset)
```

Arguments

`dataset`
: A data frame with `Coral_ID` in column 1 and single-letter allele codes in the remaining columns.

Value

A data frame of the all-`NA` colonies (with `genet` and `AdequateData` columns added), or `NULL` if there are none.

handleError_ColumnContract

Validate the required input-column contract

Description

Enforces the input format the rest of MatchMakeR relies on: column 1 must be named `Coral_ID`, column 2 must be named `MatchMaker_Index`, and every `MatchMaker_Index` value must be a whole number (no missing values). If any condition fails, the function stops with an informative message rather than silently renaming or coercing columns. `MatchMaker_Index` is used only for bookkeeping (ordering output back to the database); it is not used in any genetic calculation.

Usage

```
handleError_ColumnContract(dataset)
```

Arguments

`dataset`	A data frame read from the input CSV, before any processing.

Value

Invisibly returns `dataset` if the contract is satisfied; otherwise throws an error.

handleError_ProhibitedData

Report columns that do not hold valid SNP data

Description

Checks every column other than the `Coral_ID` and `MatchMaker_Index` keys and reports (via message) any whose values are not all valid `IUPAC` allele pairs (e.g., a site-name column or an invalid base pair). Reporting only; the offending columns are actually dropped by `convertBasePairstoCodes()`.

Usage

```
handleError_ProhibitedData(dataset, acceptableData = IUPAC)
```

Arguments

`dataset`	A data frame with `Coral_ID`, `MatchMaker_Index`, SNP data, and possibly other fields.
`acceptableData`	A reference table of allowable allele pairs (defaults to the package `IUPAC` table); must contain an `Allelepairs` column.

Value

Invisibly NULL; called for its message side effect.

---

isolateAllNAColonies *Separate colonies with no valid SNP data from the rest*

Description

Splits the input into (1) colonies that have at least one scored locus and (2) colonies that are NA at every locus. The latter are identified by handleError_allZeros() and returned with pctNull = 100 for later appending to the genet-assignment output.

Usage

```
isolateAllNAColonies(dataset)
```

Arguments

dataset	A wide data frame with a unique Coral_ID in column 1 and single-letter IUPAC allele codes in the remaining columns (as produced by convertBasePairstoCodes()).

Value

A list of two elements: [[1]] the data with all-NA colonies removed, and [[2]] a data frame of the all-NA colonies (columns Coral_ID, genet, pctNull, AdequateData), or NULL if there are none.

IUPAC *IUPAC data*

Description

A lookup table for converting 2-character alleles to single-letter IUPAC codes

Usage

```
IUPAC
```

Format

A data frame with 17 rows and 2 variables:

Allelepairs Character with two letters separated by a colon; ? for no data

IUPACallele One-letter character; ? for no data

kinshipCalcs *Calculate pairwise kinship across all comparisons*

Description

Calculates pairwise kinship for each colony pair from their allele dosages at every fully scorable locus. This function does *not* exclude invariant loci; `kinshipCalcsNoInvar()` handles that.

Usage

```
kinshipCalcs(dataset)
```

Arguments

`dataset`	A pairwise allele-match table (columns `coral1`, `coral2`, `allele1`, `allele2`, `locus`, `match`) from `determineAllAlleleMatchesOthers()`.

Value

A data frame with `coral1`, `coral2`, `totalProbLoci`, `totalScorable`, and `avg_kinship` for each colony pair.

kinshipCalcsNoInvar *Population- and individual-level kinship and gene diversity*

Description

Calculates individual- and population-level mean kinship and gene diversity (`1 - kinship`) across all colonies and loci, excluding invariant loci and within-colony comparisons. Optionally restricts the population summary to the `targetN` least related colonies by iteratively removing the colony with the highest mean kinship until `targetN` remain.

Usage

```
kinshipCalcsNoInvar(dataset, subset = FALSE, targetN = NULL)
```

Arguments

`dataset`	Wide single-letter allele data (`Coral_ID` plus one column per locus) from `convertBasePairstoCodes()`.
`subset`	Logical; if `TRUE`, also compute the `targetN`-colony summary. Requires `targetN`. Default `FALSE`.
`targetN`	Number of least-related colonies to retain when `subset = TRUE`. Must be >= 2.

Value

A list with PopAvgMKGD (population mean kinship and gene diversity), MK_init (individual mean kinship for all colonies), and MK_final (individual mean kinship for the retained colonies, or NULL when subset = FALSE).

See Also

runKinship() for notes on how clonal replicates (ramets) are handled and what PopAvgGD represents.

---

omitInvariantLoci *Remove invariant loci*

Description

Drops loci (columns) that carry only a single distinct allele across colonies, as required before kinship calculations. All-NA loci are retained (they contribute nothing to kinship because such comparisons are never fully scorable) to preserve the original behaviour.

Usage

```
omitInvariantLoci(dataset)
```

Arguments

dataset	A data frame with Coral_ID in column 1 and single-letter allele codes in the remaining columns.

Value

The input data frame with invariant loci removed.

---

readGeneticData *Read and validate a genetic SNP data file*

Description

Reads a genotype CSV of paired alleles (e.g., C:G, A:T) with Coral_ID and MatchMaker_Index as the first two columns. Values are read as character (so T is not treated as TRUE), internal whitespace is stripped, the input-column contract is enforced by handleError_ColumnContract(), and MatchMaker_Index is converted to integer. Missing SNP values are recorded as "?" so unscored allele combos can be tracked through the IUPAC translation.

Usage

```
readGeneticData(fileloc)
```

Arguments

fileloc	Path to the CSV data file. When calling this function outside runGenets() / runKinship(), supply the full path yourself, e.g. readGeneticData(file.path(getwd(), "Data", "myfile.csv")).

Value

A list of two data frames: [[1]] the cleaned genotype data (Coral_ID, MatchMaker_Index, and SNP columns), and [[2]] an index of Coral_ID and integer MatchMaker_Index.

---

returnGenetIdentity *Assign colonies to genets from clonal pairings*

Description

Given the set of colony pairs judged to be clones (matching above threshold, with adequate data), assigns each colony to a genet. Colonies are nodes and clonal pairings are edges; each connected component is one genet. A colony that clones with no other colony forms its own single-member genet. The colony graph is built directly from the coral1/coral2 columns, so colony identifiers may contain any characters (including periods).

Usage

```
returnGenetIdentity(clonePairs)
```

Arguments

clonePairs	A data frame of clone pairings with (at least) coral1 and coral2 columns, as produced within groupByGenets(). Self-pairs (coral1 == coral2) are used to register singleton colonies.

Value

A data frame with columns Coral_ID and genet (an integer genet label), one row per colony.

---

runGenets *Assign genets for every data file in the working directory*

Description

Wrapper that reads each genotype CSV in a Data folder, assigns colonies to genets, and writes a genetAssignment_<file>.csv to a Results folder (optionally also a pairwiseAlleleMatches_<file>.csv). Requires Data and Results folders in the working directory. All CSV files in Data are processed; results are keyed by input file name.

Usage

```
runGenets(
  PctMatchThreshold = NULL,
  PctNotNullThreshold = NULL,
  getPairwiseAlleleMatches = FALSE
)
```

Arguments

PctMatchThreshold	Minimum percent allele match across loci for two colonies to be called a match/clone. Required.
PctNotNullThreshold	Minimum percent of loci with valid data required to trust a comparison. Required.
getPairwiseAlleleMatches	Logical; if TRUE, also write and return the full pairwise comparison table for each file. Default FALSE.

Value

Invisibly, a list with genetAssignments (a named list of per-file genet-assignment data frames) and, when getPairwiseAlleleMatches = TRUE, pairwiseAlleleMatches (a named list of per-file pairwise tables). Each genet-assignment data frame has columns Coral_ID, MatchMaker_Index, genet, pctNull, AdequateData.

---

runKinship *Run kinship calculations for every data file in the working directory*

Description

Wrapper that reads each genotype CSV in a Data folder and computes individual- and population-level mean kinship and gene diversity (excluding invariant loci), writing results to a Results folder. Requires Data and Results folders in the working directory. All CSV files in Data are processed; results are keyed by input file name.

Usage

```
runKinship(subset = FALSE, targetN = NULL)
```

Arguments

subset	Logical; if TRUE, also compute the targetN-colony summary. Requires targetN. Default FALSE.
targetN	Number of least-related colonies to retain when subset = TRUE. Must be >= 2.

Details

Kinship and gene diversity are computed across all colonies in the input, including multiple ramets (physical fragments) of the same genet. This is intentional: every colony is treated as an individually eligible candidate (e.g., for breeding or outplanting), so clonal replicates are retained rather than collapsed to one representative per genet. Consequently, `PopAvgGD` describes the genetic diversity of the *eligible colony pool*, not a clone-corrected, genet-level population estimate. When `subset = TRUE`, the routine preferentially sheds genetically redundant colonies but keeps a representative of each clonal group. If a clone-corrected population estimate is needed instead, collapse colonies to one representative per genet (e.g., using the genet assignments from `runGenets()`) before running kinship.

Value

Invisibly, a list with `PopAvgMKGD` and `kinship_init` (each a named list of per-file data frames) and, when `subset = TRUE`, `kinship_targetN`. `kinship_*` data frames have columns `Coral_ID`, `MatchMaker_Index`, `ind_mean_kinship`.

Index