Package 'KinformR'

February 17, 2026

Title Relationship-Informed Pedigree and Variant Scoring

Version 0.1.2

Maintainer Cameron M. Nugent <cam.nugent@sequencebio.com>

Description Comparative evaluation of families and candidate variants in rare-variant association studies. The package can be used for two methodologically overlapping but distinct purposes. First, the prior to any genetic or genomic evaluation, evaluation of relative detection power of pedigrees, can direct recruitment efforts by showing which individuals not yet sampled would be the most meaningful additions to a study.
Second, after sequencing and analysis, variants based on association with disease status and familial relationships of individuals, aids in variant prioritization.
Methodology is described in
Nugent (2025) <doi:10.1101/2025.10.06.25337426>.

License MIT + file LICENSE

URL https://github.com/SequenceBio/KinformR

BugReports https://github.com/SequenceBio/KinformR/issues

Encoding UTF-8

RoxygenNote 7.3.2

VignetteBuilder knitr

Suggests devtools, testthat, knitr, rmarkdown

NeedsCompilation no

Author Cameron M. Nugent [aut, cre] (ORCID:
<https://orcid.org/0000-0002-1135-2605>)

Repository CRAN

Date/Publication 2026-02-17 15:50:02 UTC

Contents

add.fam.scores 2
assign.a 3

assign.status . . . 3
assign.u . . . 4
build.relation.dict . . . 4
calc.rv.score . . . 5
encode.rows . . . 6
ibd . . . 7
penetrance . . . 8
read.indiv . . . 8
read.pedigree . . . 9
read.relation.mat . . . 10
read.var.table . . . 10
score . . . 11
score.fam . . . 11
score.pedigree . . . 13
score.variant.status . . . 14
subset.mat . . . 15

Index **16**

`add.fam.scores` *Sum all the given scores and return a single vector with cumulative "score", "for" and "against" vals. For use in instances where one wishes to combine scores from multiple families.*

Description

Sum all the given scores and return a single vector with cumulative "score", "for" and "against" vals. For use in instances where one wishes to combine scores from multiple families.

Usage

```
add.fam.scores(score.vec)
```

Arguments

`score.vec`	A vector will all of the per family score outputs.

Value

A vector with the summed scores of all inputs.

Examples

```
score.fam1 <- c("score" = 1.0, "score.for" = 2.0, "score.against" = 1.0)
score.fam2 <- c("score" = 1.0, "score.for" = 3.0, "score.against" = 2.0)
out <- add.fam.scores(c(score.fam1, score.fam2))
```

`assign.a` *For affecteds: Take genetic variant and determine the category of the combo.*

Description

For affecteds: Take genetic variant and determine the category of the combo.

Usage

```
assign.a(variant)
```

Arguments

`variant`	Variant for individual. genotypes, phased genotypes, or binary encodings accepted.

`assign.status` *Take a disease status and a genetic variant and determine which category the combo falls in. A.c = Affected individual with ALT variant A.i = Affected individual without ALT variant U.c = Unaffected individual without ALT variant U.i = Unaffected individual with ALT variant If theoretical.max = TRUE the true variant statuses are ignored and all affected/unaffected are assigned A.c and U.c respectively. These encoding can then be used show what a family's max score would be.*

Description

Take a disease status and a genetic variant and determine which category the combo falls in. A.c = Affected individual with ALT variant A.i = Affected individual without ALT variant U.c = Unaffected individual without ALT variant U.i = Unaffected individual with ALT variant If theoretical.max = TRUE the true variant statuses are ignored and all affected/unaffected are assigned A.c and U.c respectively. These encoding can then be used show what a family's max score would be.

Usage

```
assign.status(status, variant, theoretical.max = FALSE)
```

Arguments

`status`	Disease status of an individual. A = affected, U = unaffected.
`variant`	Variant for individual. genotypes, phased genotypes, or binary encodings accepted.
`theoretical.max`	Should the theoretical maxima be returned instead of the observed values? When true, the scoring assumes correct variant-status pair for each individual. Default is FALSE.

Value

a string

Examples

```
assign.status("A", "0/1") == "A.c"
assign.status("A", "0|0") == "A.i"
assign.status("U", 1) == "U.i"
assign.status("U", "0|0") == "U.c"
```

`assign.u` *For unaffecteds: Take a genetic variant and determine the category of the combo.*

Description

For unaffecteds: Take a genetic variant and determine the category of the combo.

Usage

```
assign.u(variant)
```

Arguments

`variant`	Variant for individual. genotypes, phased genotypes, or binary encodings accepted.

`build.relation.dict` *Build dictionary with the relationships falling in the different categories for the query row.*

Description

Build dictionary with the relationships falling in the different categories for the query row.

Usage

```
build.relation.dict(mat.row, name.stat.dict, drop.unrelated = TRUE)
```

Arguments

`mat.row`	A row from a relationship matrix
`name.stat.dict`	A list with the labelled status/variant combo for each individual.
`drop.unrelated`	Should unrelated (-1) relationships be dropped? Default = TRUE.

Value

A list with the categorized relationship/variant information.

`calc.rv.score` *Calculate a relatedness-weighted score for a given rare variant.*

Description

These scores can be used to compare variants of interest within a family.

Usage

```
calc.rv.score(
  fam.list,
  affected.weight = 1,
  unaffected.weight = 0.5,
  unaffected.max = 8,
  max.err = 4
)
```

Arguments

`fam.list`	• A list with the names: 'A.c', 'A.i', 'U.c', 'U.i' respectively containing the affected correct, affected incorrect, unaffected correct and unaffected incorrect. - This can be generated with the function: score.variant.status - where each value in the dictionary is a list containing the reference and the reference's relatives as encoded based on their degree of relatedness to the reference (reference = 0, sibling/parent/offspring = 1, uncle/grandparent = 2, cousin = 3, etc.)
`affected.weight`	A coefficient to multiply the calculated A.c and A.i relatedness values by.
`unaffected.weight`	A coefficient to multiply the U.c and U.i relatedness values by.
`unaffected.max`	This param controls the score given to a first degree unaffected relatives scores decay from this specified maximum by half for each subsequent relationship degree.
`max.err`	A heuristic cap of the number of incorrect assignments allowed when scoring. When the total number of incorrect (sum of affected and unaffected) is exceeded, the variant's score is set to 0, regardless of the number of points for or against. This simplifies scoring and allows for fast filtering of poor quality variants. Default is 4.

Details

For each encoded relationship, a relationship-informed weight is applied to their sharing or not sharing of a variant. The score for affected status is: (1 / coefficient.of.relatedness) * status.weight For example, an affected cousin (encoded as a 3) would get a score of: (1/0.125) * affected weight 8 * 1 = 8 points in favour of the variant. Whereas for unaffected individuals, scores decay the

further a person is in relation to the proband based on the formula: ((unaffected.max*2) * coefficient.of.relatedness) * unaffected.weight For example, with the default unaffected.max of 8. The sister that does not have a variant would get a score of ((8*2) 0.5) * unaffected.weight (16 * 0.5) * 0.5 = 4 points for the variant. If these were the only two relatives considered we could sum the points and get a score in favour of the variant of 8 + 4 = 12 If there is evidence against a variant, this is factored into the score as: total.score = evidence.for - evidence.against For example, if there were also an affected sibling without the variant we would have the score against of: (1/0.5) * 1 = 2 The final score for the variant would then be for - against = total 12 - 2 = 10 Giving a final score of 10 for the variant. Comparing values across variants can be used to rank them based on pedigree-informed levels of variant sharing across affected and unaffected individuals. Increasing the affected.weight relative to the unaffected.weight will make the scores give more weight to the correct/incorrect status of affected individuals. The default is 2:1 weight for affected relative to unaffected, which accounts for the fact that variants are likely to be incompletely penetrant and we therefore want to be more tolerant of unaffected individuals that have a variant rather than affected individuals that do not.

Input:

Value

A list with three components: score, score.for, score.against.

Examples

```
relations <- list("A.c" = c(0, 1, 3, 1), "A.i" = c(3), "U.c" = c(1, 2), "U.i" = c(1))
rv.scores <- calc.rv.score(relations)
```

`encode.rows` *Take the relationship matrix and the encoded statuses of info. For each row, generate the encoded data for scoring.*

Description

Take the relationship matrix and the encoded statuses of info. For each row, generate the encoded data for scoring.

Usage

```
encode.rows(relation.mat, status.df, ...)
```

Arguments

`relation.mat`	The relationship matrix for all pairwise combinations of individuals.
`status.df`	The ID, status, and genotypes for each individual.
`...`	Additional arguments to be passed between methods.

Value

A dictionary with the per-individual relationship lists. One value for each row of the matrix.

ibd *Calculation of Identity by descent (IBD).*

Description

Use the relationship informationfrom the pedigree to estimate of the amount of the genome they have inherited it from a common ancestor without recombination.

Usage

```
ibd(a, b, c, d, n, K, theoretical = TRUE)
```

Arguments

a	Count of affected individuals
b	Count of obligate carriers
c	Count of children of either affecteds or carriers, with no children of their own
d	Count of Trees of unaffected individuals - specifically, two sequential generations (i.e. a parent and their offspring)
n	Count of the number of second generation progeny in a given tree.
K	The estimate of penetrance rate.
theoretical	Boolean indicating if the calculation should be theoretical IBD calculation (using only d and k), or if the calculation should use the provided n value.

Details

Can do this for the total potential relatedness in a pedigree (theoretical=TRUE), or for the actual relatedness across collected samples (theoretical=FALSE). For the theoretical=TRUE case, in the unaffected trees, if we have a sample from the parent, then the offspring do not provide any additional information for a max IBD calculation. This means that K does not scale with n.

For theoretical=FALSE, sometimes we don't have the healthy parent in an unaffected tree, and only have a child. In this case, the IBD contribution from the child is 1/4, and since they're unaffected and therefore are a counter-filter, they would contribute 1-1/4 = 3/4 to the total relatedness. Either the parent is a non-obligate carrier, or is a non-carrier. The probability of the children depends on which of those is true, so we have the additional set of terms in the theoretical=FALSE logic.

Value

pi-hat value. The proportion of genome shared between individuals.

Examples

```
ibd <- ibd(3, 1, 5, 2, 1, 0.4576484)
```

`penetrance` *Likelihood function for calculation of Pedigree-based autosomal dominant penetrance value. Formula deployed via optimize so as to determine the optimal value.*

Description

Likelihood function for calculation of Pedigree-based autosomal dominant penetrance value. Formula deployed via optimize so as to determine the optimal value.

Usage

```
penetrance(K, a, b, c, d, n)
```

Arguments

K	The range of penetrance values to be explored by the optimization function.
a	Count of affected individuals
b	Count of obligate carriers
c	Count of children of either affecteds or carriers, with no children of their own
d	Count of trees of unaffected individuals - specifically, two sequential generations (i.e. a parent and their offspring)
n	Count of the number of second generation progeny in a given tree.

Value

K Pedigree-based estimation of autosomal dominant penetrance rate.

Examples

```
K <- optimize(penetrance, c(0,1), 3, 1, 5, 2, 1, maximum=TRUE)$max
```

`read.indiv` *Read in variant and status info for individuals.*

Description

Read in variant and status info for individuals.

Usage

```
read.indiv(fname)
```

Arguments

fname	A file name, expected format of contents is: name status variant MS-5678-1001 A 0/1

Value

A data frame.

Examples

```
tsv.name1 <-system.file('extdata/1234_ex2.tsv', package = 'KinformR')
id.df1 <- read.indiv(tsv.name1)
```

---

read.pedigree *Read in the encoded pedigree data file.*

Description

Read in the encoded pedigree data file.

Usage

```
read.pedigree(filename)
```

Arguments

filename	name of the file with the data.

Value

A data frame containing the encoded pedigree information.

Examples

```
example.pedigree.file <- system.file('extdata/example_pedigree_encoding.tsv',
package = 'KinformR')
example.pedigree.df <- read.pedigree(example.pedigree.file)
```

`read.relation.mat` *Read in relationship matrix Apply the individual names to the rows and columns.*

Description

Row/column intersections give the degree of relationship for the two individuals. 0 = self, -1 = unrelated.

Usage

```
read.relation.mat(fname)
```

Arguments

`fname`	The file with the relationship matrix information.

Value

A matrix with the relationships and individual ids as rownames and colnames.

Examples

```
mat.name1 <-system.file('extdata/1234_ex2.mat', package = 'KinformR')
mat1 <- read.relation.mat(mat.name1)
```

`read.var.table` *Read in a vcf-like subset of information obtained from use of seqbiopy's vcf_extract function on a vcf with the status encoded in the indivudal's names*

Description

Note - ensure the status in the names match your desired encoding! There are individuals with ambiguous statues, that you may require to be encoded in a specific fashion for you current purposes.

Usage

```
read.var.table(fname)
```

Arguments

`fname`	A file name, expected format of contents is: #CHROM POS REF ALT MS-5678-1001_A MS-5678-1002_U ... chr3 46203838 G A 0/1 0/0 ...

Value

A dataframe. Data will be worked into a data frame with format. name status variant MS-5678-1001 A 0/1

Examples

```
ex.infile <-system.file('extdata/example_vcf_extract_5678.tsv',
                        package = 'KinformR')
read.var.table(ex.infile)
```

---

score *Score the pedigrees using the pihat values.*

Description

Score the pedigrees using the pihat values.

Usage

```
score(pihat)
```

Arguments

pihat	Estimated proportion of genome shared between individuals, from function: ibd.

Value

The score value.

Examples

```
s.val <- score(12.61)
```

---

score.fam *Given a relationship matrix and status dataframe, score a family by applying the calc.rv.score scoring system to every pairwise combination of individuals.*

Description

By default all individuals are treated as the reference 'proband' and the given variant's score is calculated based on relationships to all other individuals. e.g. for each row in the relationship matrix. calc.rv.score is run, with the row name indicating the reference individual that the calculation is relative to. Note that the relation.mat can include more individuals than are present within the status.df, but the matrix will be subset to include only those individuals that have status information provided.

Usage

```
score.fam(
  relation.mat,
  status.df,
  affected.weight = 1,
  unaffected.weight = 0.5,
  return.sums = FALSE,
  return.means = TRUE,
  affected.only = TRUE,
  max.err = 4
)
```

Arguments

relation.mat	A relationship matrix for the family.
status.df	A dataframe with the encoded variant/disease status of each individual
affected.weight	A coefficient to multiply the calculated A.c and A.i relatedness values by.
unaffected.weight	A coefficient to multiply the U.c and U.i relatedness values by.
return.sums	Boolean indicating if sum of family variant scores should be returned (default = FALSE).
return.means	Boolean indicating if mean of all family variant scores should be returned (default = TRUE).
affected.only	Boolean indicating if family score should be calculated using only affected individuals (default = TRUE).
max.err	A heuristic cap of the number of incorrect assignments allowed when scoring. When the total number of incorrect (sum of affected and unaffected) is exceeded, the variant's score is set to 0, regardless of the number of points for or against. This simplifies scoring and allows for fast filtering of poor quality variants. Default is 4.

Details

There are several return options possible.

- If affected.only is TRUE, the final scores will be reported for only rows where the reference individual is affected (default = True).
- If return.means is TRUE, the average scores for the rows will be reported. (default = TRUE)
- If return.sums is True, the sum of the scores for all the rows will be reported. (default = False) NOTE: if affected.only = True, the averages and sums are calculated using only the affected reference individuals.

Value

A labelled vector with names: score, score.for, score.against

Examples

```
mat.name1 <- system.file("extdata/1234_ex2.mat", package = "KinformR")
tsv.name1 <- system.file("extdata/1234_ex2.tsv", package = "KinformR")
mat.df <- read.relation.mat(mat.name1)
ind.df <- read.indiv(tsv.name1)
ind.df.status <- score.variant.status(ind.df)
score.default <- score.fam(mat.df, ind.df.status)
```

`score.pedigree` *Take the encoded information about the pedigrees and calculate penetrance.*

Description

Determine a value score of families by comparing their relationship structure. More distant relationships between affecteds (e.g. affected cousins) is more valuable that close relationships (e.g. affected siblings) as there is less IBD and therefore a smaller search space.

Usage

```
score.pedigree(h)
```

Arguments

`h`	A data frame containing the encoded pedigree information

Details

Simplifying assumptions:

- Autosomal dominant
- No ambiguous statuses
- No more than two sequential generations of unknown carrier status (non-obligate carrier vs. non-carrier). Generalized support of arbitrary tree structures gets a lot more complicated, especially for the likelihood function.
- Exclude big giant trees of unaffecteds - related to above. Will slightly bias the result toward higher penetrance.
- Exclude subjects younger than age of onset

Value

A data frame containing the theoretical scoring of the power of a family assuming you were able to collect everyone on the simplified pedigree, as well as a current scoreing, examining only those for whom you currently have DNA.

Examples

```
example.pedigree.file <-system.file('extdata/example_pedigree_encoding.tsv',
package = 'KinformR')
example.pedigree.df <- read.pedigree(example.pedigree.file)
penetrance.df <- score.pedigree(example.pedigree.df)
```

`score.variant.status` *Take the dataframe with variants and status and determine which indivudals are scored correctly and which are scored incorrectly. Assign an A.c, A.i, U.c, U.i, unk*

Description

Variants can be encoded as binary (0 or 1, genotypes 0/0 or 0/1, or phased genotypes 0|0 0|1). Note the program assumes alt is the disease allele. homozygous alts are allowed.

Usage

```
score.variant.status(indiv.df, theoretical.max = FALSE)
```

Arguments

`indiv.df`	A dataframe with the format: name status variant MS-5678-1001 A 0/1
`theoretical.max`	Should the theoretical maxima be returned instead of the observed values? When true, the scoring assumes correct variant-status pair for each individual. Default is FALSE.

Details

theoretical.max - bool, default is FALSE when TRUE, function encodes the theoretical max, using a dummy perfect associatng variant generated to see what a family could score. TODO - switch to numbers 1-4 and -1?

Value

Copy of input dataframe, with dataframe with the status categroies added as a new column "statvar.cat"

`subset.mat` *Take the matrix and subset out only the encoded individuals that are present in the status dataframe.*

Description

Take the matrix and subset out only the encoded individuals that are present in the status dataframe.

Usage

```
## S3 method for class 'mat'
subset(mat.df, status.df)
```

Arguments

`mat.df`	The full matrix file to subset
`status.df`	The list of sampled individuals, matrix is subset to only these individuals.

Value

A subset of the input matrix.

Index

add.fam.scores, 2
assign.a, 3
assign.status, 3
assign.u, 4

build.relation.dict, 4

calc.rv.score, 5

encode.rows, 6

ibd, 7

penetrance, 8

read.indiv, 8
read.pedigree, 9
read.relation.mat, 10
read.var.table, 10

score, 11
score.fam, 11
score.pedigree, 13
score.variant.status, 14
subset.mat, 15