KinformR - pedigree-informed rare variant association scoring

Cameron M. Nugent

12 February, 2026

Introduction

When looking at rare variants within a family, not all affected and unaffected individuals are of equal importance. `KinformR` is an R package meant to aid in comparative evaluation of information in family-based rare-variant association studies; the package considers evidence of rare variant's association with disease status in a family and scores the variant based on relationships of individuals in a pedigree. Comparing scores across candidates can thereby help in assessing their relative merit.

The program leverages Wright's coefficient of relatedness to score families based on the relationship of individuals as well as their disease status and genotypes for a given variant. A candidate variant is considered strongest when it is shared by an affected individual and other distantly-related affected family members and not shared by closely related unaffected family members. The theory behind this is that more distantly related individuals have a smaller proportion of their genome that is IBD, so the chance of a false positive (shared variant through chance, not due to association with disease) is lower. Conversely, when a closely related relative is unaffected, and the affected and unaffected individuals have different genotypes for the candidate variant, this provides evidence that the disease is likely not associated with the large IBD portion of the genome shared by the two individuals and thereby gives evidence in favour of the candidate.

Load the library

```r
library(KinformR)

show <- function(df){
  knitr::kable(df, format = "markdown", digits = 2)
}
```

Input data

Two input are required to score a candidate variant: the family relationship information and the variant status for the individuals.

The relation matrix

This input is a matrix containing all the pairwise relationships of individuals in a family. The row and column names are the individual IDs, and the intersecting value denotes the degree of relationship between the individuals (self = 0, 1st degree relations = 1, etc. Unrelated individuals are given a value of -1). As of version `0.1.0` the relation matrix is a manually created file, where relationship values are assigned via manual inspection of the family pedigree.

To read in the data, one uses the function `read.relation.mat`.

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

	MS-1234-1001	MS-1234-1002	MS-1234-1003	MS-1234-1004	MS-1234-1005	MS-1234-1006	MS-1234-5001	MS-1234-6001
MS-1234-1001	0	1	1	3	3	1	1	2
MS-1234-1002	1	0	1	3	3	1	2	1
MS-1234-1003	1	1	0	3	3	1	2	2
MS-1234-1004	3	3	3	0	-1	3	4	4
MS-1234-1005	3	3	3	-1	0	3	4	4
MS-1234-1006	1	1	1	3	3	0	2	2
MS-1234-5001	2	4	4	2	2	1	0	3
MS-1234-6001	3	2	4	4	2	1	2	0

The status file

This file includes the same individual IDs used in the relationship matrix as well as the disease and variant status for all individuals.

Note that the order of individuals can be different between the two files. All individual in the status file must be present within the relationship matrix, but you can have individuals in the relationship matrix that are not present in the status file. The program assumes there is no genotype information for individuals not in the status file and they are ignored in the variant scoring.

To read in the data, one uses the function `read.indiv`.

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

show(status.df)
```

name	status	variant
MS-1234-1001	A	0/1
MS-1234-1002	U	0/1
MS-1234-1003	A	0/1
MS-1234-1004	A	0/1
MS-1234-1005	A	0/0
MS-1234-1006	U	0/0
MS-1234-5001	A	0/1
MS-1234-6001	U	0/0

The disease-genotype scoring can then be encoded using the `score.variant.status` function to produce the status-variant category for all individuals. This creates a df with the new column: `statvar.cat`.

```r
full.df.status <-  score.variant.status(status.df)
show(full.df.status)
```

name	status	variant	statvar.cat
MS-1234-1001	A	0/1	A.c
MS-1234-1002	U	0/1	U.i
MS-1234-1003	A	0/1	A.c
MS-1234-1004	A	0/1	A.c
MS-1234-1005	A	0/0	A.i
MS-1234-1006	U	0/0	U.c
MS-1234-5001	A	0/1	A.c
MS-1234-6001	U	0/0	U.c

Scoring a family

For most real-world applications, you will likely want to score family members in conjunction with one another, and take the mean score for an entire family. This can be accomplished with `score.fam`, which takes in the matrix of relationships and the table with encoded `statvar.cat` of all individuals.

```r
ex.score.default <- score.fam(rel.mat, full.df.status)
show(ex.score.default)
```

	x
score	21.3
score.for	27.6
score.against	6.3

By default `score.fam` returns: - The scores considering only the Affected individuals as the reference individual (skipping the rows for the U individuals: MS-1234-1002, MS-1234-1006, and MS-1234-6001, in the previous example). - The mean of the calculated score from each reference individuals.

As previously noted, if an individual is present in the relationship matrix and not in the status file, it is assumed there is no genetic information for this individual and they are ignored when calculating the variant score.

The scoring can be changed to summing across all combinations as opposed to the mean by passing the following options. Note using the program in this way will return higher scores for more dense pedigrees.

```r
ex.score.sum <- score.fam(rel.mat, full.df.status,
                          return.sums = TRUE, return.means = FALSE)
show(ex.score.sum)
```

	x
score	106.5
score.for	138.0
score.against	31.5

To obtain a long form table with the scores for variants expressed relative to each individual, set both `return.sums` and `return.means` to `FALSE`. This output can aid in identifying which individuals are carrying the most weight in a family's score.

```r
ex.score.table <- score.fam(rel.mat, full.df.status,
                            return.sums = FALSE, return.means = FALSE)
show(ex.score.table)
```

	score	score.for	score.against
MS-1234-1001	7.0	19.0	12.0
MS-1234-1003	9.0	21.0	12.0
MS-1234-1004	33.5	34.5	1.0
MS-1234-1005	31.5	33.5	2.0
MS-1234-5001	25.5	30.0	4.5

How scoring works

A Minimal example, scoring a variant from perspective of a single individual.

This section is meant to demonstrate how the variant scoring is accomplished on a finer scale. A user does not need to interact with the package on this level of granularity. This section is for explanatory purposes only, demonstrating how the `score.fam` function operated "under the hood".

The `score.fam` function runs the scoring method once for each affected individual in the status dataframe (or for each individual regardless of status if `affected.only = FALSE`). To do this, for each individual, the program takes corresponding row of the relationship matrix to determine the relations to all other individuals in the pedigree.

For example, the degrees of relationships of all other members of the example family relative to the reference individual `"MS-1234-1001"` are show in the following subset of the matrix:

```r
rel.mat.proband <-  rel.mat["MS-1234-1001",]
show(rel.mat.proband)
```

	x
MS-1234-1001	0
MS-1234-1002	1
MS-1234-1003	1
MS-1234-1004	3
MS-1234-1005	3
MS-1234-1006	1
MS-1234-5001	1
MS-1234-6001	2

This is taken along with a list of encoded statuses of all individuals for the given variant. Above `score.variant.status` used the disease status and a genetic variant of each individual to determine which of the following categories they fall in to: - A.c = Affected individual with variant - A.i = Affected individual without variant - U.c = Unaffected individual without variant - U.i = Unaffected individual with variant Within `score.fam`, a labelled list of encodings for all individuals is generated.

```r
name.stat.dict <- full.df.status$statvar.cat
names(name.stat.dict) <- full.df.status$name
name.stat.dict
```

```
## MS-1234-1001 MS-1234-1002 MS-1234-1003 MS-1234-1004 MS-1234-1005 MS-1234-1006
##        "A.c"        "U.i"        "A.c"        "A.c"        "A.i"        "U.c"
## MS-1234-5001 MS-1234-6001
##        "A.c"        "U.c"
```

`build.relation.dict` is used summarize the collate the status and relationship information and evidence affected and unaffected relations giving evidence for and evidence against a variant.

```
rel.dict<-build.relation.dict(rel.mat.proband, name.stat.dict)
rel.dict
```

```
## $A.c
## [1] 0 1 3 1
##
## $A.i
## [1] 3
##
## $U.c
## [1] 1 2
##
## $U.i
## [1] 1
```

In this example, the proband, two first degree relations, and a third degree relations are all affected and share the candidate variant. For the affected correct (`A.c`) category we therefore see the following encoded:

```
rel.dict$A.c
```

```
## [1] 0 1 3 1
```

Since one first degree unaffected relative has the variant, they are categorized as "unaffected incorrect"(`U.i`) and we see:

```
rel.dict$U.i
```

```
## [1] 1
```

Deriving a relatedness-weighted score for the variant from the perspective of the given individual is then performed by `calc.rv.score`

For each degree-encoded relationship, the coefficient of relatedness is used to weight the evidence for or against a variant. The coefficients for different degrees of relationship are:

```
for(i in 0:7){
    print(paste0("Degree of relatedness: ", i,
                 " coefficient of relatedness: ",  1 / (2 ** (i))))
}
```

```
## [1] "Degree of relatedness: 0 coefficient of relatedness: 1"
## [1] "Degree of relatedness: 1 coefficient of relatedness: 0.5"
## [1] "Degree of relatedness: 2 coefficient of relatedness: 0.25"
## [1] "Degree of relatedness: 3 coefficient of relatedness: 0.125"
## [1] "Degree of relatedness: 4 coefficient of relatedness: 0.0625"
## [1] "Degree of relatedness: 5 coefficient of relatedness: 0.03125"
## [1] "Degree of relatedness: 6 coefficient of relatedness: 0.015625"
## [1] "Degree of relatedness: 7 coefficient of relatedness: 0.0078125"
```

The score for affected status increase as individuals become more distantly related. The formula is:

```
   (1 / coefficient.of.relatedness) * affected.weight
```

For example, an affected cousin (encoded as a 3 and having a coefficient of relatedness of 0.125) would get a score of:

```
     (1/0.125) * affected.weight
     8 * 1
     = 8 points in favour of the variant.
```

For unaffected individuals, scores decay the further a person is in relation to the query individual based on the formula:

```
((unaffected.max*2) * coefficient.of.relatedness ) * unaffected.weight
```

For example, with the default `unaffected.weight = 0.5` and unaffected 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.

This is all accomplished by the function `calc.rv.score`.

```r
calc.rv.score(rel.dict)
```

```
## $score
## [1] 7
##
## $score.for
## [1] 19
##
## $score.against
## [1] 12
```

The weights of the scoring can be adjusted, for example if we wanted to consider only `affected`-based evidence, we could turn off the unaffected part of the calculation by setting the unaffected weighting to 0. This can be useful for incompletely penetrant variants, where disease status and genotype of unaffected individuals are more likely to have imperfect concordance.

Additionally, families with low numbers of affected individuals sequenced and high number of unaffected individuals may haved inflated variant scores and potentially be misleading, focusing the scoring algorithm on the affected individuals only can overcome this bias.

```r
calc.rv.score(rel.dict, unaffected.weight=0)
```

```
## $score
## [1] 5
##
## $score.for
## [1] 13
##
## $score.against
## [1] 8
```

The `score.fam` function automatically walks through this process from all specified perspectives in the pedigree and by default returns the average score. The use of the averages and different perspectives is meant to eliminate pedigree-associated bias, such as for instances when a proband is distantly related to all other members in a family (considering the relationships from only the perspective of the proband in this case would give an inflated score for the variant's value).