KinformR - penetrance and idb informed scoring of families

Cameron M. Nugent

12 February, 2026

Introduction

The `cal.penetrance` and associated `penetrance` and `ibd` functions facilitate comparison of the relative "power" of families in a study. This can be accomplished upstream of sequencing (i.e. in the project planning stage) and is therefore only dependent on relationship structure and reported affected status of individuals in a given family or set of families.

Estimating power of families

The `cal.penetrance` function generates both a theoretical ranking of the power of a family assuming you were able to collect everyone on the simplified pedigree, as well as a current ranking, examining only those for whom you currently have DNA. This allows evaluation of the impact on the ranking if certain other family members are enrolled in the study.

Load the library

```
library(KinformR)

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

The input data

The family power calculations depend on a single tab-delimited input file, where each row represents a family. The input file is read in using the `read.pedigree` function.

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

example.pedigree.df <- read.pedigree(example.pedigree.file)
```

The input file is expected to have the following 11 columns (with a header).

```
colnames(example.pedigree.df)
```

```
##  [1] "Family" "max_a"  "a"      "max_b"  "b"      "max_c"  "c"      "max_d"
##  [9] "d"      "max_n"  "n"
```

Simplified summary of pedigrees

For now this file should be be constructed through careful manual inspection of the predigrees. To encode the rows for each family, you should first prune down pedigrees to informative allele transfers. For the purposes of this tool, we exclude young generations (non-adults, younger than age of onset) and large (more than two sequential generations) trees of exclusively unaffected family members. Additionally all individuals require a binary A/U status, there should be no ambiguous individuals. There will be some judgment calls required here.

Encoding categories of relationships

From the simplified pedigrees, the individuals are assigned to the following categories.

Category	Description
a	Affected individuals
b	Obligate carriers
c	Children of either affecteds or carriers, with no children of their own
d	Trees of unaffected individuals - specifically, two sequential generations (i.e. a parent and their offspring; trees of unaffecteds that are larger than this are omitted.)

The counts of individuals assigned to these categories are then added to the tab-delimited input file:

```
show(example.pedigree.df)
```

Family	max_a	a	max_b	b	max_c	c	max_d	d	max_n	n
1111	4	4	0	0	0	0	0	0	0	0
1122	1	1	0	0	0	0	1	1	2	2
1133	1	1	1	1	0	0	1	1	1	1
1144	1	1	0	0	1	1	1	1	1	1
1155	1	1	0	0	1	1	0	0	0	0
1166	1	1	0	0	0	0	1	1	0	0
2222	4	4	0	0	0	0	1	1	2	2
0347	6	2	6	3	17	1	7,3,1,1	1	2,3,1,8	1
5031	5	2	8	4	2	4	5	2	0	0
6325	7	2	5	0	4	0	1,2	0	1,2	0
1234	5	4	4	1	4	0	1,2	1	1,3	1
9876	7	4	2	0	7	4	1	1	5	1
4006	2	2	1	0	4	1	0	0	0	0
3734	3	0	2	0	3	0	1,3,1	1	1,3,5	0

All columns with the prefix `max_` are meant to count the total number of each category in the pedigree, while the columns without this prefix are the number of each category for whom samples have been collected.

The categories correspond to A, B, and C as defined above.

Category D is represented by two numbers, d and n. n is the number of offspring in a tree of unaffecteds; d is the number of those types of trees across the pedigree. Multiple types of trees are encoded with commas separating the values. For example, the following represents a family with three total trees of unaffecteds. One tree (d=1) has three offspring (n=3); two trees (d=2) each have one offspring (n=1).

```
d   n
1,2 3,1
```

Theoretical vs. current rankings

With the encoded data loaded, the function `cal.penetrance` will generate both a theoretical ranking of the power of a family assuming you were able to collect everyone on the simplified pedigree, as well as a current ranking, examining only those for whom you currently have DNA. This allows evaluation of the impact on the ranking if certain other family members are enrolled in the study.

```
penetrance.df <- score.pedigree(example.pedigree.df)

show(penetrance.df)
```

family	penetrance	max.pi-hat	max.score	current.pi-hat	current.score	pct.of.max
1111	1.00	3.00	2.08	3.00	2.08	100.00
1122	1.00	2.00	1.39	1.25	0.87	62.50
1133	0.41	2.33	1.61	2.00	1.38	85.73
1144	0.70	3.24	2.25	2.83	1.96	87.50
1155	1.00	2.00	1.39	2.00	1.39	100.00
1166	1.00	2.00	1.39	2.00	1.39	100.00
2222	1.00	5.00	3.47	4.25	2.95	85.00
0347	0.21	44.49	30.84	6.02	4.17	13.53
5031	0.32	20.71	14.36	12.47	8.64	60.19
6325	0.48	20.78	14.40	1.00	0.69	4.81
1234	0.42	17.39	12.05	5.01	3.47	28.79
9876	0.64	20.45	14.17	10.39	7.20	50.82
4006	0.45	7.48	5.18	2.37	1.64	31.69
3734	0.32	14.01	9.71	0.25	0.17	1.80

The output includes seven columns that with the following information:

Output Column	Description
family	The family id.
penetrance	Estimated penetrance rate (K) for the family.
max_pi-hat	The estimated proportion of the genome that is shared between all individuals *that could be sampled* in the family (IBD).
max_score	The theoretical maximum score for the given family *that could* be achieved if all individuals were sampled.
current_pi-hat	The estimated proportion of the genome that is shared between all individuals *that have been sampled* in the family (IBD).
current_score	The score for the given family *that has* been achieved through the sampled individuals
pct_of_max	The percentage of the theoretical maximum score that has been realized in the family sampling.

With the scoring completed, the values can be queried to learn more about the realized and potential value of the families relative to one another.

Sorting on `current_score` shows which family has the most detection power based on collected samples.

```r
ord.df.current <- penetrance.df[order(penetrance.df$current.score, decreasing = TRUE),]
show(ord.df.current)
```

	family	penetrance	max.pi-hat	max.score	current.pi-hat	current.score	pct.of.max
9	5031	0.32	20.71	14.36	12.47	8.64	60.19
12	9876	0.64	20.45	14.17	10.39	7.20	50.82
8	0347	0.21	44.49	30.84	6.02	4.17	13.53
11	1234	0.42	17.39	12.05	5.01	3.47	28.79
7	2222	1.00	5.00	3.47	4.25	2.95	85.00
1	1111	1.00	3.00	2.08	3.00	2.08	100.00
4	1144	0.70	3.24	2.25	2.83	1.96	87.50
13	4006	0.45	7.48	5.18	2.37	1.64	31.69
5	1155	1.00	2.00	1.39	2.00	1.39	100.00
6	1166	1.00	2.00	1.39	2.00	1.39	100.00
3	1133	0.41	2.33	1.61	2.00	1.38	85.73
2	1122	1.00	2.00	1.39	1.25	0.87	62.50
10	6325	0.48	20.78	14.40	1.00	0.69	4.81
14	3734	0.32	14.01	9.71	0.25	0.17	1.80

If we had to work with only what we have, then family 5031 gives the most detection power.

Sorting on `max_score` shows which family could have the most detection power, if all individuals were sampled. This can be useful in targeting future sampling efforts as it shows where more samples would give the most value.

```r
ord.df.max <- penetrance.df[order(penetrance.df$max.score, decreasing = TRUE),]
show(ord.df.max)
```

	family	penetrance	max.pi-hat	max.score	current.pi-hat	current.score	pct.of.max
8	0347	0.21	44.49	30.84	6.02	4.17	13.53
10	6325	0.48	20.78	14.40	1.00	0.69	4.81
9	5031	0.32	20.71	14.36	12.47	8.64	60.19
12	9876	0.64	20.45	14.17	10.39	7.20	50.82
11	1234	0.42	17.39	12.05	5.01	3.47	28.79
14	3734	0.32	14.01	9.71	0.25	0.17	1.80
13	4006	0.45	7.48	5.18	2.37	1.64	31.69
7	2222	1.00	5.00	3.47	4.25	2.95	85.00
4	1144	0.70	3.24	2.25	2.83	1.96	87.50
1	1111	1.00	3.00	2.08	3.00	2.08	100.00
3	1133	0.41	2.33	1.61	2.00	1.38	85.73
2	1122	1.00	2.00	1.39	1.25	0.87	62.50
5	1155	1.00	2.00	1.39	2.00	1.39	100.00
6	1166	1.00	2.00	1.39	2.00	1.39	100.00

Here we can see that family `0347` has a maximum score of 30.84, but a realized score of only 4.17. Given the high potential detection power for this family, it is an ideal target for future sampling efforts as current samples reveal on 13.5% of the family's potential value.

Note on a few special cases

In trees of unaffecteds, there are two special cases for current ranking: 1. You have collected the parent (regardless of collection status of the child). In this case, the child cannot provide any additional information beyond the parent, so we only count the parent. ($d=1$, $n=0$; equivalently, $c=1$) 2. You have collected one or more children, but not the parent. In this case, each of the children contribute a portion of what the parent would have contributed to our understanding. ($d=1$, $n>0$)