pedgene package vignette
Version 3.9

Jason Sinnwell and Daniel Schaid

September 8, 2024

1 Introduction

This document is a brief tutorial for the pedgene package, which implements methods described in Schaid et al., 2013 [1], entitled *"Multiple Genetic Variant Association Testing by Collapsing and Kernel Methods With Pedigree or Population Structured Data"*.

We begin by showing the data structure and requirements of the input data, and continue with examples on running the method. If the pedgene package is not loaded, load it now.

```
> library(pedgene)
```

2 Example Data

Three example data frames are provided within the pedgene package:

- *example.ped:* data.frame with columns ped, person, father, mother, sex, and trait; for 3 families of 10 subjects each, and 9 unrelated subjects
- *example.geno:* data.frame with ped and person to match lines to example.ped, and minor allele count for 20 simulated variant positions
- *example.map:* data frame with columns for gene name and chromosome matching columns in example.geno

Load the example data and look at some of the values.

```
> data(example.ped)
> example.ped[c(1:10,31:39),]
```

```
     famid person father mother sex trait
1        1      1      0      0   1    NA
2        1      2      0      0   2    NA
3        1      3      0      0   2    NA
4        1      4      1      2   1    NA
5        1      5      1      2   2    NA
6        1      6      0      0   1    NA
7        1      7      4      3   2     1
8        1      8      4      3   1     1
9        1      9      6      5   2    NA
10       1     10      6      5   2     1
1942   592      1      0      0   1     0
1943   593      1      0      0   1     0
1944   594      1      0      0   1     0
1945   595      1      0      0   1     0
```

```
1946   596      1      0      0   1     0
1947   597      1      0      0   1     0
1948   598      1      0      0   1     0
1949   599      1      0      0   1     0
1950   600      1      0      0   1     0
```

```
> data(example.geno)
> dim(example.geno)
```

```
[1] 18 22
```

```
> example.geno[,c(1:2,10:14)]
```

```
   famid person AA.8 AA.9 AA.10 AX.1 AX.2
7      1      7    0    0     0    0    0
8      1      8    0    0     0    0    0
10     1     10    0    0     0    0    0
17     2      7    0    0     0    0    0
18     2      8    0    0     0    0    1
20     2     10    0    0     0    0    0
27     3      7    0    0     0    0    0
28     3      8    0    0     0    0    0
30     3     10    0    0     0    0    0
31   592      1    0    0     0    0    0
32   593      1    0    0     0    0    0
33   594      1    0    0     0    0    0
34   595      1    0    0     0    0    0
35   596      1    0    0     0    0    0
36   597      1    1    0     0    0    0
37   598      1    0    0     0    0    1
38   599      1    0    0     0    0    0
39   600      1    1    0     1    0    0
```

```
> data(example.map)
> example.map
```

```
   chrom gene
1      1   AA
2      1   AA
3      1   AA
4      1   AA
5      1   AA
6      1   AA
7      1   AA
8      1   AA
9      1   AA
10     1   AA
11     X   AX
12     X   AX
13     X   AX
14     X   AX
15     X   AX
16     X   AX
17     X   AX
18     X   AX
```

```
19      X   AX
20      X   AX
```

Note the following features:

- The pedigree data has more subjects than subjects that have genotypes, to keep pedigrees connected.
- Subjects with genotype data are matched to the pedigree data by the ped and person IDs in both data sets
- The number of non-id columns in the genotype data frame must match the number of rows in the map data frame
- The two genes provided are actually the same simulated data, but we show they are analyzed differently for chromosome 1 versus chromosome X

3 Case-Control

No covariate adjustment

We perform a typical use of pedgene with default settings on the case-control status provided in the example data.

```
> pg.cc <- pedgene(ped=example.ped, geno=example.geno, map=example.map)
> print(pg.cc)

  gene chrom n.variant n.noninform stat.kernel pval.kernel stat.burden pval.burden
1   AA     1         7           3    144.8264   0.3498380   -2.047518  0.04060727
2   AX     X         7           3    124.0772   0.2037578   -2.167566  0.03019169
```

The results show the output for the two genes in a table, containing gene, chromosome, the number of variants used, the non-informative variants that were removed, the kernel statistic and p-value, and the burden statistic and p-value. The kernel p-value is calculated using Kounen's saddlepoint method [2], and the burden p-value is based on the normal distribution.

With covariate adjustment

The pedgene function can work with case-control data and an adjusted trait using a covariate. The data contains the sex variable, so we use it in a logistic regression to get adjusted fitted values for the trait, which we add to example.ped at a new column named *trait.adjusted*, which pedgene recognizes and uses in calculating residuals.

```
> binfit <-  glm(trait ~ (sex-1),data=example.ped, family="binomial")
> example.ped$trait.adjusted[!is.na(example.ped$trait)] <- fitted.values(binfit)
> example.ped[1:10,]

   famid person father mother sex trait trait.adjusted
1      1      1      0      0   1    NA             NA
2      1      2      0      0   2    NA             NA
3      1      3      0      0   2    NA             NA
4      1      4      1      2   1    NA             NA
5      1      5      1      2   2    NA             NA
6      1      6      0      0   1    NA             NA
7      1      7      4      3   2     1      0.6322153
8      1      8      4      3   1     1      0.5673055
9      1      9      6      5   2    NA             NA
10     1     10      6      5   2     1      0.6322153

> pg.cc.adj <- pedgene(ped=example.ped, geno=example.geno, map=example.map)
> summary(pg.cc.adj, digits=4)
```

```
Summary for pedgene object:

Call:
pedgene(ped = example.ped, geno = example.geno, map = example.map)


  gene chrom n.variant n.noninform stat.kernel pval.kernel stat.burden pval.burden
1   AA     1         7           3       189.2      0.1855      -2.599    0.009352
2   AX     X         7           3       160.6      0.1298      -2.583    0.009792
```

The results are now shown from the summary method, which adds the call to the pedgene function. The results are different because we now input the fitted values from the logistic regression, which of course can be a more complex model than what we use here.

4 Continuous Trait

No covariate adjustment

The pedgene method works similarly for continous traits. We first create a different version of the pedigree data to have a continous trait, which we simulate. We simulate complete trait data for all subjects, but pedgene will only use the subjects for which we have genotype data.

```
> set.seed(10)
> cont.ped <- example.ped[,c("famid", "person", "father", "mother", "sex")]
> beta.sex <- 0.3
> cont.ped$trait <- (cont.ped$sex-1)*beta.sex + rnorm(nrow(cont.ped))
> pg.cont <- pedgene(ped = cont.ped, geno = example.geno, map = example.map)
> print(pg.cont, digits=4)

  gene chrom n.variant n.noninform stat.kernel pval.kernel stat.burden pval.burden
1   AA     1         7           3       92.16      0.8305      0.1971      0.8437
2   AX     X         7           3       58.08      0.7775      0.2271      0.8204
```

The results are now non-significant because we simulated a trait that is slightly influenced by sex, but not the genotype data.

With covariate adjustment

Now we add the *trait.adjusted* column to the cont.ped data set and run pedgene again.

```
> gausfit <-  glm(trait ~ (sex-1),data=cont.ped, family="gaussian")
> cont.ped$trait.adjusted <- fitted.values(gausfit)
> cont.ped[1:10,]

   famid person father mother sex       trait trait.adjusted
1      1      1      0      0   1  0.01874617     -0.1619874
2      1      2      0      0   2  0.11574746     -0.3239749
3      1      3      0      0   2 -1.07133055     -0.3239749
4      1      4      1      2   1 -0.59916772     -0.1619874
5      1      5      1      2   2  0.59454513     -0.3239749
6      1      6      0      0   1  0.38979430     -0.1619874
7      1      7      4      3   2 -0.90807618     -0.3239749
8      1      8      4      3   1 -0.36367602     -0.1619874
9      1      9      6      5   2 -1.32667268     -0.3239749
10     1     10      6      5   2  0.04352161     -0.3239749
```

```
> pg.cont.adj <- pedgene(ped=cont.ped, geno=example.geno, map=example.map)
> summary(pg.cont.adj, digits=5)

Summary for pedgene object:

Call:
pedgene(ped = cont.ped, geno = example.geno, map = example.map)


  gene chrom n.variant n.noninform stat.kernel pval.kernel stat.burden pval.burden
1   AA     1         7           3     143.586     0.72428    -1.07780     0.28112
2   AX     X         7           3      88.507     0.68011    -0.88632     0.37544
```

5 Remarks

- The pedgene package is deigned to be rigid in what it expects for pedigree and genotype data, which puts a little extra burden on the user. However it should reduce confusion when running the method.
- One potential run-time bottleneck is a set of pedigree checks performed on all pedigrees. If the pedigrees are clean and you have many subjects, you may wish to skip this step with the argument checkpeds=FALSE.
- By default, the method uses Madsen-Browning weights [4], but it allows user-defined weights per variant with the weights argument.

6 R Session Information

```
> toLatex(sessionInfo())
```

- R version 4.4.1 (2024-06-14), `x86_64-pc-linux-gnu`
- Locale: `LC_CTYPE=en_US.UTF-8`, `LC_NUMERIC=C`, `LC_TIME=en_US.UTF-8`, `LC_COLLATE=C`, `LC_MONETARY=en_US.UTF-8`, `LC_MESSAGES=en_US.UTF-8`, `LC_PAPER=en_US.UTF-8`, `LC_NAME=C`, `LC_ADDRESS=C`, `LC_TELEPHONE=C`, `LC_MEASUREMENT=en_US.UTF-8`, `LC_IDENTIFICATION=C`
- Time zone: `Etc/UTC`
- TZcode source: `system (glibc)`
- Running under: `Ubuntu 24.04.1 LTS`
- Matrix products: default
- BLAS: `/usr/lib/x86_64-linux-gnu/openblas-pthread/libblas.so.3`
- LAPACK: `/usr/lib/x86_64-linux-gnu/openblas-pthread/libopenblasp-r0.3.26.so` ; LAPACK version3.12.0
- Base packages: base, datasets, grDevices, graphics, grid, methods, stats, utils
- Other packages: CompQuadForm 1.4.3, Matrix 1.7-0, kinship2 1.9.6.1, pedgene 3.9, quadprog 1.5-8, survey 4.4-2, survival 3.7-0
- Loaded via a namespace (and not attached): DBI 1.2.3, Rcpp 1.0.13, buildtools 1.0.0, compiler 4.4.1, knitr 1.48, lattice 0.22-6, maketools 1.3.0, mitools 2.4, splines 4.4.1, sys 3.4.2, tools 4.4.1, xfun 0.47

References

[1] Schaid DJ, McDonnell SK., Sinnwell JP, Thibodeau SN. (2013) **Multiple Genetic Variant Association Testing by Collapsing and Kernel Methods With Pedigree or Population Structured Data** *Genet Epidemiol*, 37(5):409-18.

[2] Kounen D (1999) **Saddle point approximatinos for distributions of quadratic forms in normal variables** *Biometrika*, 86:929 -935.

[3] Davies RB (1980) **Algorithm AS 155: The Distribution of a Linear Combination of chi-2 Random Variables** *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 29(3):323-33

[4] Madsen BE, Browning SR (2009). **A groupwise association test for rare mutations using a weighted sum statistic** *PLoS Genet* 5(2):e1000384.