# Package 'RVS'

April 1, 2025

**Type** Package

**Title** Computes estimates of the probability of related individuals sharing a rare variant

**Version** 1.28.0

**Date** 2024-02-14

**Author** Alexandre Bureau, Ingo Ruczinski, Samuel Younkin, Thomas Sherman

**Maintainer** Alexandre Bureau <alexandre.bureau@fmed.ulaval.ca>

**Description** Rare Variant Sharing (RVS) implements tests of association and linkage between rare genetic variant genotypes and a dichotomous phenotype, e.g. a disease status, in family samples. The tests are based on probabilities of rare variant sharing by relatives under the null hypothesis of absence of linkage and association between the rare variants and the phenotype and apply to single variants or multiple variants in a region (e.g. gene-based test).

**License** GPL-2

**Depends** R (>= 3.5.0)

**Imports** GENLIB, gRain, snpStats, kinship2, methods, stats, utils, R.utils

**Suggests** knitr, testthat, rmarkdown, BiocStyle, VariantAnnotation

**VignetteBuilder** knitr

**LazyData** true

**RoxygenNote** 7.1.0

**Encoding** UTF-8

**biocViews** ImmunoOncology, Genetics, GenomeWideAssociation, VariantDetection, ExomeSeq, WholeGenome

**Collate** 'RVgene.R' 'pedigree-methods.R' 'RVsharing.R' 'documentation.R' 'grainNetworkHelper.R' 'monteCarloMethods.R' 'multipleFamilyCalculations.R' 'multipleFamilyCalculationsBackend.R' 'relatedFoundersCorrection.R' 'sharingProbabilityCalculations.R' 'sharingProbabilityCalculationsSplitting.R'

**git_url** https://git.bioconductor.org/packages/RVS

**git_branch** RELEASE_3_20

**git_last_commit** b74c74d

**git_last_commit_date** 2024-10-29

**Repository** Bioconductor 3.20

**Date/Publication** 2025-03-31

# Contents

## `ancestorDistance` *distance between a descendant and an ancestor*

### Description

distance between a descendant and an ancestor

### Usage

```
ancestorDistance(procPed, a, d)
```

### Arguments

| | |
|---|---|
| `procPed` | pedigree that has been through `processPedigree` |
| `a` | ancestor subject |
| `d` | descendant subject |

### Value

minimum distance (number generations) between a and d

## `areMating` *determine if two subjects have a child together*

### Description

determine if two subjects have a child together

### Usage

```
areMating(procPed, f1, f2)
```

### Arguments

| | |
|---|---|
| `procPed` | pedigree that has been through `processPedigree` |
| `f1` | subject 1 |
| `f2` | subject 2 |

### Value

true if both subjects share a child

## checkArgs *check arguments provided to RVsharing for validty*

### Description

verifies that arguments are valid, throws an error if they are not

### Usage

```
checkArgs(alleleFreq, kinshipCoeff, nSim, founderDist)
```

### Arguments

| | |
|---|---|
| alleleFreq | allele frequency among the founders |
| kinshipCoeff | mean kinship coefficient among the founders |
| nSim | number of simulations used in monte carlo calculation |
| founderDist | custom distribution among founders. Only used when simulating probability with nSim |

### Value

throws error if arguments invalid

## ComputeKinshipPropCoef *ratio of excess kinship among descendants over mean kinship among founders*

### Description

Computes, for each pair of final descendants in the pedigree structure contained in the pedigree object, the ratio of the difference between the inferred and expected kinship coefficient for the pair over the mean kinship among founders.

### Usage

```
ComputeKinshipPropCoef(ped)

## S4 method for signature 'pedigree'
ComputeKinshipPropCoef(ped)
```

### Arguments

| | |
|---|---|
| ped | pedigree object (S3) |

## Details

The ratio for each pair of final descendants is computed using equation (A1) of Bureau et al. Dividing the difference between the inferred and expected kinship coefficient for each pair by this ratio gives a pair-specific estimate of the mean kinship among founders, which can then be averaged over all pairs of final descendants from the same population to obtain a global estimate of the mean kinship among founders.

## Value

a symmetric matrix of ratios for all pair of final descendants in the pedigree structure contained in the pedigree

## References

Bureau, A., Younkin, S., Parker, M.M., Bailey-Wilson, J.E., Marazita, M.L., Murray, J.C., Mangold, E., Albacha-Hejazi, H., Beaty, T.H. and Ruczinski, I. (2014) Inferring rare disease risk variants based on exact probabilities of sharing by multiple affected relatives. Bioinformatics, 30(15): 2189-96, doi:10.1093/bioinformatics/btu198.

## Examples

```
data(samplePedigrees)
ComputeKinshipPropCoef(samplePedigrees$firstCousinTriple)
```

---

`computePFU` *computation of P[FjU] using equation 21 of Bureau et al.*

---

## Description

computation of P[FjU] using equation 21 of Bureau et al.

## Usage

```
computePFU(nf, theta, ord = 5)
```

## Arguments

| | |
|---|---|
| `nf` | number of founders of the pedigree |
| `theta` | value of the parameter of the polynomial distribution |
| `ord` | order of the polynomial approximation to the distribtion of the number of distinct alleles in the founders (noted d in Bureay et al.). Must be <= 5 |

## Value

P[FjU] (scalar)

## `computePhiVec` *expected kinship coefficient for different number of alleles*

### Description

expected kinship coefficient for different number of alleles

### Usage

```
computePhiVec(nf, amin = 2 * nf - 2)
```

### Arguments

| | |
|---|---|
| `nf` | number of founders |
| `amin` | minimum number of distinct alleles |

### Value

vector of expected phi_a for nf founders for numbers of distinct alleles from amin to 2*nf-1

## `convertMatrix` *convert snpMatrix to a list of vectors of sharing*

### Description

convert snpMatrix to a list of vectors of sharing

### Usage

```
convertMatrix(snpMat, famIds, minorAllele)
```

### Arguments

| | |
|---|---|
| `snpMat` | SnpMatrix |
| `famIds` | family ids corresponding to rows of the snpMap |
| `minorAllele` | vector specifying the minor allele of each variant |

### Value

list of boolean vectors indicating sharing pattern for each variant

## `createNetwork` *create bayesian network from processed pedigree*

### Description

Creates a bayesian network using the gRain package. The network is built based on the information in a pedigree object that has been processed using `processPedigree`.

### Usage

```
createNetwork(procPed, prior = c(1, 2, 1))
```

### Arguments

| | |
|---|---|
| `procPed` | processed Pedigree object |
| `prior` | prior on number of alleles for founders |

### Value

bayesian network from gRain package

## `denomProb` *denominator of sharing probability*

### Description

calculates the denominator of the sharing probability outline in section 2.1 of Bureau et al.

### Usage

```
denomProb(net, procPed)
```

### Arguments

| | |
|---|---|
| `procPed` | pedigree object that has been process with processPedigree |
| `gRain` | bayesian network |

### Value

denominator value

## `enrichmentPValue` *enrichment p-value across multiple families and variants*

### Description

Computes a p-value for all variants seen across all families

### Usage

```
enrichmentPValue(snpMat, famInfo, sharingProbs, threshold = 0)
```

### Arguments

| | |
|---|---|
| `snpMat` | SnpMatrix |
| `famInfo` | data frame containing pedigree, member, father, mother, sex, affected fields for each sequenced subject |
| `sharingProbs` | vector of sharing probabilites, must be a named vector with famid's for each probability |
| `threshold` | minimum p-value threshold passed to multipleFamilyPValue |

### Details

For each variant, the families which have all sequenced subjects sharing the variant and the families which have some sequenced subjects sharing the variant are recorded. All unique (family, variant) pairs are accumulated into a single vector and passed to multipleFamilyPValue

### Value

p-value

### References

Fu, J., Beaty, T.H., Scott, A.F., Hetmanski, J., Parker, M.M., Bailey-Wilson, J.E., Marazita, M.L., et al. 2017. Whole Exome Association of Rare Deletions in Multiplex Oral Cleft Families. Genetic Epidemiology 41 (1): 61–69. doi:10.1002/gepi.22010.

## `enrichmentPValue_R_Backend` *R backend for enrichmentPValue calculation*

### Description

R backend for enrichmentPValue calculation

### Usage

```
enrichmentPValue_R_Backend(
  snpMat,
  famIds,
  sharingProbs,
  minorAllele,
  threshold = 0
)
```

### Arguments

| | |
|---|---|
| `snpMat` | SnpMatrix |
| `famIds` | family ids corresponding to rows of the snpMap |
| `sharingProbs` | vector of sharing probabilites, must be a named vector with famid's for each probability |
| `minorAllele` | which variant value to count as the minor allele |
| `threshold` | minimum p-value threshold passed to multipleFamilyPValue |

### Value

p-value

---

## `ex.ped.mat` *matrix of pedigree information and genotype data from famVCF stored in the LINKAGE format*

---

### Description

matrix of pedigree information and genotype data from famVCF stored in the LINKAGE format

### Usage

```
ex.ped.mat
```

---

## `exactSharingProb` *exact sharing probability calculation*

---

### Description

Calculate the exact sharing probability given the minor allele frequency among the founders (population).

### Usage

```
exactSharingProb(procPed, alleleFreq)
```

### Arguments

| | |
|---|---|
| `procPed` | pedigree that has been through processPedigree() |
| `alleleFreq` | allele frequency among the founders |

### Value

sharing probability

## `extract_carriers` *extract carriers of minor allele*

### Description

finds the carriers of the minor allele at a specified site

### Usage

```
extract_carriers(ped, site, fam, type = "alleles", minor.allele = 2)
```

### Arguments

| | |
|---|---|
| `ped` | pedigree coded in a ped file with either two alleles per variant ("alleles"), or a count of one allele ("count") |
| `site` | site where to record carriers |
| `fam` | ID of the family for which to extract carriers |
| `type` | representation of allele count |
| `minor.allele` | id of minor allele |

### Value

carriers in ped

## `fam15157.vcf` *VCF objects containing genotype data for two families: fam15157 and fam28003 (corresponding to the secondCousinTriple and firstAndSecondCousinsTriple families in samplePedigrees)*

### Description

VCF objects containing genotype data for two families: fam15157 and fam28003 (corresponding to the secondCousinTriple and firstAndSecondCousinsTriple families in samplePedigrees)

### Usage

```
fam15157.vcf
```

---

`fam28003.vcf` *VCF objects containing genotype data for two families: fam15157 and fam28003 (corresponding to the secondCousinTriple and firstAndSecondCousinsTriple families in samplePedigrees)*

---

## Description

VCF objects containing genotype data for two families: fam15157 and fam28003 (corresponding to the secondCousinTriple and firstAndSecondCousinsTriple families in samplePedigrees)

## Usage

```
fam28003.vcf
```

---

`founderOccurence` *determine if subjects are descended from founders*

---

## Description

determine if subjects are descended from founders

## Usage

```
founderOccurence(procPed, subjects, founders)
```

## Arguments

| | |
|---|---|
| `procPed` | pedigree that has been through `processPedigree` |
| `subjects` | vector of subject ids |
| `founders` | vector of founder ids |

## Value

data frame with 0/1 for if a subject if descended from founder

## GeneDrop *deprecated function*

### Description

This function is deprecated with version >= 2.0 and should not be used, instead use RVsharing with nSim option

### Usage

```
GeneDrop(...)

GeneDropSim.allsubsets.fn(...)

GeneDropSim.fn(...)

GeneDropSimExcessSharing.fn(...)
```

### Arguments

| | |
|---|---|
| `...` | arguments to the old function |

### Value

none

### Examples

```
tryCatch(GeneDrop(), error = function(e) message(e))
```

## get.psubset *deprecated function*

### Description

This function is deprecated with version >= 2.0 and should not be used, instead use multipleFamilyPValue

### Usage

```
get.psubset(vec, not, pshare.data)
```

### Arguments

| | |
|---|---|
| `vec` | a vector of names of all families where a variant is seen |
| `not` | a vector of names of families where not all affected subjects share the rare variant |
| `pshare.data` | a data frame with at least two of the following columns: pshare: vector of RV sharing probabilities ped.tocompute.vec: vector of names of the families whose sharing probability is contained in pshare. The names in the arguments vec and not must be found in ped.tocompute.vec |

## Value

P-value of the exact rare variant sharing test requiring sharing by all affected subjects.

## Examples

```
data(samplePedigrees)
notSharedFams <- c(15159, 15053, 15157)
famids <- sapply(samplePedigrees, function(p) p$famid[1])
notShared <- famids %in% notSharedFams
probs <- sapply(samplePedigrees, RVsharing)
get.psubset(famids, notShared, data.frame(pshare=probs,
ped.tocompute.vec=famids))
```

---

## `inferNumAlleles` *most likely number of distinct alleles among founders*

---

## Description

Calculates the most likely number of distinct alleles among nf founders based on the mean estimated kinship coefficient

## Usage

```
inferNumAlleles(phi, nf)
```

## Arguments

| | |
|---|---|
| `phi` | mean estimated kinship coefficient |
| `nf` | number of founders |

## Value

number of distinct alleles

---

## `inferTheta` *solve the parameter theta for polynomial approximation of the distribution of the number of distinct alleles.*

---

## Description

solve the parameter theta for polynomial approximation of the distribution of the number of distinct alleles.

## Usage

```
inferTheta(phi, phiVec)
```

## Arguments

| | |
|---|---|
| `phi` | the mean estimated kinship between founders |
| `phiVec` | contains phi_a for a = 2*nf-ord to 2*nf-1, where ord must be between 2 and 5 |

### Value

real roots of the polynomial approximation

---

## `isDescendant` *determine if one subject is a descendant of another*

---

### Description

determine if one subject is a descendant of another

### Usage

```
isDescendant(procPed, a, d)
```

### Arguments

| | |
|---|---|
| `procPed` | pedigree that has been through `processPedigree` |
| `a` | ancestor subject |
| `d` | descendant subject |

### Value

true if d is descended from a

---

## `marginalProb` *calculates the marginal probability of a set of nodes*

---

### Description

Given a bayesian network from the gRain package and a named list of (nodes, states), this function returns the joint-marginal probability of each node taking a value in the specified set of states.

### Usage

```
marginalProb(net, states)
```

### Arguments

| | |
|---|---|
| `net` | bayesian network from gRain package |
| `states` | named list of states for each node |

### Details

This function calculates the probability P(A,B,C) by factoring it into conditional probabilities, i.e. P(A|B,C) * P(B|C) * P(C). Starting at the right side, P(C) is computed and then evidence of C being true is added to the network and P(B) is computed - effectively giving the probability P(B|C). This process continues from right to left until the entire product has been computed.

### Value

joint-marginal probability

## `monteCarloSharingProb` *calculates sharing probability by simulating pedigree outcomes*

### Description

Calculates the same exact probability as RVsharing, except uses monte carlo simulation instead of exact computation. This method allows for more flexibility in the scenarios considered.

### Usage

```
monteCarloSharingProb(
  procPed,
  alleleFreq = NA,
  kinshipCoeff = NA,
  nSim,
  founderDist = NULL,
  kinshipOrder = 5
)
```

### Arguments

| | |
|---|---|
| `procPed` | pedigree that has been through `processPedigree` |
| `alleleFreq` | allele frequency among the founders |
| `kinshipCoeff` | mean kinship coefficient among the founders |
| `nSim` | number of simulations used in monte carlo calculation |
| `founderDist` | custom distribution among founders. Only used when simulating probability with nSim |
| `kinshipOrder` | order of the polynomial approximation to the distribtion of the number of distinct alleles in the founders (d in Bureau et al.). Must be <= 5 |

### Value

sharing probability between all carriers in pedigree

## `multipleFamilyPValue` *probability of sharing of rare variants in a subset of families*

### Description

Computing probability of sharing of rare variants in a subset of families where rare variants are seen based on precomputed family-specific rare variant sharing probabilities.

### Usage

```
multipleFamilyPValue(sharingProbs, observedSharing, minPValue = 0)
```

### Arguments

| | |
|---|---|
| `sharingProbs` | named vector of sharing probabilties, where names correspond to famid value of pedigree |
| `observedSharing` | boolean vector describing if all affected subjects in the family share the variant (TRUE if all share) |
| `minPValue` | the minimum p-value threshold, once the true p-value is determined to be less than this, the computation stops and minPValue is returned - this prevents extremely long computations for extremely small p-values |

### Details

All the subsets of families of size equal or inferior to the length of not are created, and the joint probability of each such subset not sharing a rare variant and the remaining families sharing a rare variant is obtained as the product of the family-specific rare variant sharing probabilities or its complement. The function then sums the pattern probabilities inferior or equal to the probability of the observed pattern of the not families not sharing a rare variant and the remaining families sharing a rare variant.

### Value

P-value of the exact rare variant sharing test requiring sharing by all affected subjects

### References

Bureau, A., Younkin, S., Parker, M.M., Bailey-Wilson, J.E., Marazita, M.L., Murray, J.C., Mangold, E., Albacha-Hejazi, H., Beaty, T.H. and Ruczinski, I. (2014) Inferring rare disease risk variants based on exact probabilities of sharing by multiple affected relatives. Bioinformatics, 30(15): 2189-96, doi:10.1093/bioinformatics/btu198.

### Examples

```
data(samplePedigrees)
probs <- sapply(samplePedigrees, RVsharing)
notSharedFams <- c(15159, 15053, 15157)
famids <- sapply(samplePedigrees, function(p) p$famid[1])
shared <- !famids %in% notSharedFams
names(shared) <- names(probs)
multipleFamilyPValue(probs, shared)
```

---

## `multipleFamilyPValue_R_Backend` *R backend for multipleFamilyPValue calculation*

---

### Description

R backend for multipleFamilyPValue calculation

### Usage

```
multipleFamilyPValue_R_Backend(sharingProbs, observedSharing, minPValue = 0)
```

### Arguments

| | |
|---|---|
| `sharingProbs` | named vector of sharing probabilties, where names correspond to famid value of pedigree |
| `observedSharing` | boolean vector describing if all affected subjects in the family share the variant (TRUE if all share) |
| `minPValue` | the minimum p-value threshold, once the true p-value is determined to be less than this, the computation stops and minPValue is returned - this prevents extremely long computations for extremely small p-values |

### Value

p-value

---

## `multipleVariantPValue` *generalization of multipleFamilyPValue to multiple variants*

---

### Description

Computes a p-value for each variant sharing pattern across families

### Usage

```
multipleVariantPValue(
  snpMat,
  famInfo,
  sharingProbs,
  minorAllele = NULL,
  filter = NULL,
  alpha = 0
)
```

### Arguments

| | |
|---|---|
| `snpMat` | SnpMatrix |
| `famInfo` | data frame containing pedigree, member, father, mother, sex, affected fields for each sequenced subject |
| `sharingProbs` | vector of sharing probabilites, must be a named vector with famid's for each probability |
| `minorAllele` | vector specifying the minor allele of each variant |
| `filter` | criteria for filtering pvalues |
| `alpha` | parameter for filter |

### Details

For each variant, the families which have all sequenced subjects sharing the variant and the families which have some sequenced subjects sharing the variant are recorded. These values are passed to multipleFamilyPValue

### Value

list containing p-values and potential p-values for each variant

---

## `multipleVariantPValue_R_Backend` *R backend for multipleVariantPValue calculation*

---

### Description

R backend for multipleVariantPValue calculation

### Usage

```
multipleVariantPValue_R_Backend(
  snpMat,
  famIds,
  sharingProbs,
  minorAllele,
  filter = NULL,
  alpha = 0
)
```

### Arguments

| | |
|---|---|
| `snpMat` | SnpMatrix |
| `famIds` | family ids corresponding to rows of the snpMap |
| `sharingProbs` | vector of sharing probabilites, must be a named vector with famid's for each probability |
| `minorAllele` | vector specifying the minor allele of each variant |
| `filter` | criteria for filtering pvalues |
| `alpha` | parameter for filter |

### Value

list of p-values and potential p-values

---

## `numerProb` *numerator of sharing probability*

---

### Description

calculates the numerator of the sharing probability outline in section 2.1 of Bureau et al.

### Usage

```
numerProb(net, procPed)
```

### Arguments

| | |
|---|---|
| `procPed` | pedigree object that has been process with processPedigree |
| `gRain` | bayesian network |

### Value

numerator value

---

## `oldArgs` *check for arguments in v1.7 format*

---

### Description

check arguments provided in ... to see if the user called RVsharing using a function signature from v1.7, this will convert the arguments into a pedigree suitable for the signature in version > 2.0

### Usage

```
oldArgs(ped, data, dad.id, mom.id)
```

### Arguments

| | |
|---|---|
| `ped` | a pedigree object |
| `data` | numeric/character vector of subject ids |
| `dad.id` | numeric/character vector of father ids, founders' parents should be NA or 0 |
| `mom.id` | numeric/character vector of mother ids, founders' parents should be NA or 0 |

### Value

if old arguments are provided, a pedigree object is returned, otherwise ped is returned

---

## `oneFounderSharingProb` *calculate sharing probability in basic case*

---

### Description

Assume that only one founder can introduce the variant to the pedigree. Condition on each founder and sum over all resulting probabilities.

### Usage

```
oneFounderSharingProb(procPed)
```

### Arguments

| | |
|---|---|
| `procPed` | pedigree that has been through processPedigree() |

### Value

sharing probability

## `oneFounderSharingProbSplitting` *calculate sharing probability in basic case*

### Description

Assume that only one founder can introduce the variant to the pedigree. Condition on each founder and sum over all resulting probabilities.

### Usage

```
oneFounderSharingProbSplitting(procPed, useFounderCouples = TRUE)
```

### Arguments

| | |
|---|---|
| `procPed` | pedigree that has been through processPedigree() |
| `useFounderCouples` | a logical value indicating whether to exploit the interchangeability of the mother and father from founder couples to save computations. Warning! This works only when all founders have only one spouse. Set to FALSE if at least one founder has two or more spouses. |

### Value

sharing probability

## `ped2trio` *deprecated function*

### Description

This function is deprecated with version >= 2.0 and should not be used.

### Usage

```
ped2trio(...)
```

### Arguments

| | |
|---|---|
| `...` | arguments to the old function |

### Value

none

### Examples

```
tryCatch(ped2trio(), error = function(e) message(e))
```

## processPedigree *extract useful information from a pedigree*

### Description

Extract key information from a pedigree object, which makes subsequent computations much easier.

### Usage

```
processPedigree(ped, carriers = NULL)

## S4 method for signature 'pedigree'
processPedigree(ped, carriers = NULL)
```

### Arguments

| | |
|---|---|
| `ped` | pedigree object (S3) |
| `carriers` | subjects in which the rare variant is seen |

### Value

list containing relevant pedigree info

### Examples

```
data(samplePedigrees)
processPedigree(samplePedigrees$firstCousinPair)
```

## relatedFoundersCorrection *make the neccesary correction for when founders have a non-zero kinship coefficient*

### Description

make the neccesary correction for when founders have a non-zero kinship coefficient

### Usage

```
relatedFoundersCorrection(nf, kinshipCoeff, ord = 5)
```

### Arguments

| | |
|---|---|
| `nf` | number of founders |
| `kinshipCoeff` | mean kinship coefficient among all founders |
| `ord` | order of the polynomial approximation to the distribtion of the number of distinct alleles in the founders (noted d in Bureay et al.). Must be <= 5 |

### Value

weight used in probability calculation

## `runMonteCarlo` *run the monte carlo simulation*

### Description

Given a number of simulations and a distribution of variants in the founders, this function simulates possbile outcomes of the pedigree and returns a sharing probability.

### Usage

```
runMonteCarlo(procPed, founderDist, nSim)
```

### Arguments

| | |
|---|---|
| procPed | pedigree that has been through `processPedigree` |
| founderDist | custom distribution among founders. Only used when simulating probability with nSim |
| nSim | number of simulations used in monte carlo calculation |

### Details

If the number of simulations is greater than 20,000 then the computation is done in parallel (as long as the package parallel is available)

### Value

sharing probability between all carriers in pedigree

## `RVgene` *Probability of sharing of rare variants in a family sample within a gene*

### Description

Computing probability of sharing of rare variants in a family sample within a genomic region such as a gene.

### Usage

```
RVgene(
  data,
  ped.listfams,
  sites,
  fams,
  pattern.prob.list,
  nequiv.list,
  N.list,
  type = "alleles",
  minor.allele.vec,
  precomputed.prob = list(0),
  maxdim = 1e+09,
  partial.sharing = TRUE,
  ...
)
```

**Arguments**

| | |
|---|---|
| `data` | A list of `SnpMatrix` objects corresponding to each pedigree object in ped.listfams, or a data.frame or matrix encoding the pedigree information and genotype data in the standard LINKAGE ped format or the PLINK raw format with additive component only (see PLINK web site [1]). From the pedigree information, only the family ID in the first column, the subject ID in the second column and the affection status in the sixth column are used (columns 3 to 5 are ignored). Also, family members without genotype data do not need to appear in this object. The genotype of each variant can be coded in two ways, each corresponding to a different value of the type option: a minor allele count on one column with missing values coded NA, (type="count") or the identity of the two alleles on two consecutive columns, with missing values coded 0 corresponding to the standard LINKAGE ped format (type="alleles"). If you provide a `SnpMatrix` object then the genotype should be coded as the minor allele count + 1, i.e. 01 is the homozygous genotype for the common allele. |
| `ped.listfams` | a list of `pedigree` objects, one object for each pedigree for which genotype data are included in `data`. |
| `sites` | a vector of the column indices of the variant sites to test in `data`. If the argument fams is provided, the variant sites are tested in each corresponding family in the fams vector (a variant present in multiple families must then be repeated for every families where it appears). |
| `fams` | an optional character vector of the names of families in `data` and `ped.listfams` carrying the variants listed in the corresponding position in `sites`. If missing, the names of the families carrying the minor allele at each position in `sites` are extracted from `data` |
| `pattern.prob.list` | a list of precomputed rare variant sharing probabilities for all possible sharing patterns in the families in `data` and `ped.listfams` |
| `nequiv.list` | an optional vector of the number of configurations of rare variant sharing by the affected subjects corresponding to the same pattern and probability in `pattern.prob.list`. Default is a vector of 1s |
| `N.list` | a vector of the number of affected subjects sharing a rare variant in the corresponding pattern in `pattern.prob.list` |
| `type` | an optional character string taking value "alleles" or "count". Default is "alleles" |
| `minor.allele.vec` | an optional vector of the minor alleles at each site in the `sites` vector. It is not needed if type="count". If it is missing and type="alleles", the minor allele is assumed to take the value 2 |
| `precomputed.prob` | an optional list of vectors precomputed rare variant sharing probabilities for families in `data` and `ped.listfams`. If the vectors are named, the names must be strings formed by the concatenation of the sorted carrier names separated by semi-columns. If the vectors are not named, the vectors must represent probabilities for all the possible values of `N.list` for the corresponding family (one probability per value of `N.list`) |

| | |
|---|---|
| `maxdim` | upper bound on the dimension of the array containing the joint distribution of the sharing patterns for all families in fams (to avoid running out of memory) |
| `partial.sharing` | logical indicating whether the test allowing for sharing by a subset of affected subjects should be performed. If FALSE, only the test requiring sharing by all affected subjects is computed. Default is TRUE |
| `...` | other arguments to be passed to RVsharing |

## Details

The function extracts the carriers of the minor allele at each entry in sites in each family where it is present in ped.mat (or in the families specified in fams if that argument is specified). It then computes exact rare variant sharing probabilities in each family for each variant by calling `RVsharing`. If multiple rare variants are seen in the same family, the smallest sharing probability among all rare variants is retained. The joint rare variant sharing probability over all families is obtained as the product of the family-specific probabilities. The p-value of the test allowing for sharing by a subset of affected subjects over the rare variants in the genomic region is then computed as the sum of the probabilities of the possible combinations of sharing patterns among all families with a probability less than or equal to the observed joint probability and a total number of carriers greater than or equal to the sum of the number of carriers in all families, using the values in `pattern.prob.list`, `nequiv.list` and `N.list`. The families where all affected subjects share a rare variant are determined by verifying if the length of the carrier vector equals the maximum value of `N.list` for that family. The p-value of the test requiring sharing by all affected subjects is computed by calling `multipleFamilyPValue`.

## Value

A list with items: `p` P-value of the exact rare variant sharing test allowing for sharing by a subset of affected subjects. `pall` P-value of the exact rare variant sharing test requiring sharing by all affected subjects. `potentialp` Minimum achievable p-value if all affected subjects were carriers of a rare variant.

## References

Bureau, A., Begum, F., Taub, M.A., Hetmanski, J., Parker, M.M., Albacha-Hejazi, H., Scott, A.F., et al. (2019) Inferring Disease Risk Genes from Sequencing Data in Multiplex Pedigrees Through Sharing of Rare Variants. Genet Epidemiol. 43(1):37-49. doi: 10.1002/gepi.22155.

## Examples

```
data(samplePedigrees)
data(ex.ped.mat)
fam15157 <- samplePedigrees$secondCousinTriple
fam15157.pattern.prob = c(RVsharing(fam15157,carriers=c(15,16,17)),
    RVsharing(fam15157,carriers=c(15,16)),
    RVsharing(fam15157,carriers=c(15)))
fam15157.nequiv = c(1,3,3)
# check that distribution sums to 1
sum(fam15157.pattern.prob*fam15157.nequiv)
fam15157.N = 3:1
fam28003 <- samplePedigrees$firstAndSecondCousinsTriple
fam28003.pattern.prob = c(RVsharing(fam28003,carriers=c(36,104,110)),
    RVsharing(fam28003,carriers=c(36,104)),
    RVsharing(fam28003,carriers=c(104,110)),
    RVsharing(fam28003,carriers=c(36)),
    RVsharing(fam28003,carriers=c(104)))
fam28003.N = c(3,2,2,1,1)
fam28003.nequiv = c(1,2,1,1,2)
# check that distribution sums to 1
sum(fam28003.pattern.prob*fam28003.nequiv)
# Creating lists
ex.pattern.prob.list = list("15157"=fam15157.pattern.prob,"28003"=fam28003.pattern.prob)
ex.nequiv.list = list("15157"=fam15157.nequiv,"28003"=fam28003.nequiv)
ex.N.list = list("15157"=fam15157.N,"28003"=fam28003.N)
ex.ped.obj = list(fam15157,fam28003)
names(ex.ped.obj) = c("15157","28003")
sites = c(92,119)
minor.allele.vec=c(1,4)
RVgene(ex.ped.mat,ex.ped.obj,sites,
    pattern.prob.list=ex.pattern.prob.list,
nequiv.list=ex.nequiv.list,N.list=ex.N.list,
    minor.allele.vec=minor.allele.vec)
# calling with a SnpMatrix list
data(famVCF)
fam15157.snp = suppressWarnings(VariantAnnotation::genotypeToSnpMatrix(fam15157.vcf))
fam28003.snp = suppressWarnings(VariantAnnotation::genotypeToSnpMatrix(fam28003.vcf))
ex.SnpMatrix.list = list(fam15157=fam15157.snp$genotypes,fam28003=fam28003.snp$genotypes)
RVgene(ex.SnpMatrix.list,ex.ped.obj,sites,
    pattern.prob.list=ex.pattern.prob.list, nequiv.list=ex.nequiv.list,
    N.list=ex.N.list,minor.allele.vec=minor.allele.vec)
```

---

`RVS` *RVS*

---

## Description

Rare Variant Sharing (RVS) implements tests of association and linkage between rare genetic variant genotypes and a dichotomous phenotype, e.g. a disease status, in family samples. The tests are based on probabilities of rare variant sharing by relatives under the null hypothesis of absence of linkage and association between the rare variants and the phenotype and apply to single variants or multiple variants in a region (e.g. gene-based test).

---

`RVsharing` *probability of sharing a rare variant among relatives*

---

## Description

computing probability that a rare variant is shared by a set of subjects in a pedigree using the gRain package

## Usage

```
RVsharing(
  ped,
  carriers = NULL,
  alleleFreq = NA,
  kinshipCoeff = NA,
  nSim = NA,
  founderDist = NULL,
  useAffected = FALSE,
  kinshipOrder = 5,
  splitPed = FALSE,
  useFounderCouples = TRUE,
  ...
)

## S4 method for signature 'pedigree'
RVsharing(
  ped,
  carriers = NULL,
  alleleFreq = NA,
  kinshipCoeff = NA,
  nSim = NA,
  founderDist = NULL,
  useAffected = FALSE,
  kinshipOrder = 5,
  splitPed = FALSE,
  useFounderCouples = TRUE,
  ...
)

## S4 method for signature 'list'
RVsharing(
  ped,
  carriers = NULL,
  alleleFreq = NA,
  kinshipCoeff = NA,
  nSim = NA,
  founderDist = NULL,
  useAffected = FALSE,
  kinshipOrder = 5,
  splitPed = FALSE,
  useFounderCouples = TRUE,
  ...
)
```

## Arguments

| | |
|---|---|
| `ped` | S3 pedigree object or a list of pedigree objects |
| `carriers` | subjects in pedigree that have the variant, if ped is a list, then this will also be a list of vectors specifying the carriers in each pedigree |
| `alleleFreq` | allele frequency among the founders |

| | |
|---|---|
| `kinshipCoeff` | mean kinship coefficient among the founders |
| `nSim` | number of simulations used in monte carlo calculation |
| `founderDist` | custom distribution among founders. Only used when simulating probability with nSim |
| `useAffected` | a logical value indicating whether to condition on seeing the variant among the affected subjects instead of the final descendants |
| `kinshipOrder` | order of the polynomial approximation to the distribtion of the number of distinct alleles in the founders (d in Bureau et al.). Must be <= 5 |
| `splitPed` | a logical value indicating whether to split the pedigree in subpedigrees below each founder to enable computations in pedigrees too large to be stored in a single Bayesian network |
| `useFounderCouples` | a logical value indicating whether to exploit the interchangeability of the mother and father from founder couples to save computations. Warning! This works only when all founders have only one spouse. Set to FALSE if at least one founder has two or more spouses. Only used when splitPed = TRUE |
| `...` | allows for additional arguments |

## Details

the function RVsharing computes the probability that all subjects identified as carriers of a rare variant in the vector carriers (or all final descendants in the pedigree if carriers == NULL) share that rare variant AND the final descendants not included in carriers do not carry it, given that the rare variant has been detected in any subject in the union of the carriers and the final descendants of the pedigree. A final descendant is defined as a subject without descendant in the pedigree, it it not necessarily in the youngest generation. If carriers enumerates a subset of pedigree members, the function will then compute the probability these carriers share the rare variant AND the final descendants not included in carriers do not carry it based on the above terms. To obtain the probability that a set of pedigree members carry a rare variant given it was seen in any of the set members (ignoring the carrier status of final descendants not in the set), the pedigree must be trimmed of the other final descendants before calling RVsharing.

## Value

sharing probability between all carriers in pedigree, or if splitPed = TRUE, a vector of sharing probabilities for all subsets of the carriers

## References

Bureau, A., Younkin, S., Parker, M.M., Bailey-Wilson, J.E., Marazita, M.L., Murray, J.C., Mangold, E., Albacha-Hejazi, H., Beaty, T.H. and Ruczinski, I. (2014) Inferring rare disease risk variants based on exact probabilities of sharing by multiple affected relatives. Bioinformatics, 30(15): 2189-96, doi:10.1093/bioinformatics/btu198.

Sherman, T., Fu, J., Scharpf, R., Bureau, A., and Ruczinski, I. (2018) Detection of rare disease variants in extended pedigrees using RVS. Bioinformatics, 1-3, doi: 10.1093/bioinformatics/bty976

## Examples

```
data("samplePedigrees")
RVsharing(samplePedigrees$firstCousinPair)
```

## samplePedigrees *list of 8 sample pedigree objects*

### Description

list of 8 sample pedigree objects

### Usage

```
samplePedigrees
```

## simulatePedigree *simulates pedigree given founder states*

### Description

Given the states (number of allele copies) of the founders, this function simulates mendelian inheritance and returns the states of all subjects in the pedigree

### Usage

```
simulatePedigree(procPed, states)
```

### Arguments

| | |
|---|---|
| `procPed` | pedigree that has been through processPedigree() |
| `states` | state of each founder (0,1,2 copies of variant) |

### Value

states for all subjects in pedigree

## snpMat *SnpMatrix with genotype information from famVCF for fam15157*

### Description

SnpMatrix with genotype information from famVCF for fam15157

### Usage

```
snpMat
```

## SnpMatrixToCount — *convert a list of SnpMatrices to a single matrix in a similiar format as LINKAGE except with minor allele counts*

### Description

creates a matrix in LINKAGE format using pedigree information from a list of pedigree objects and genotype information from a list of SnpMatrices

### Usage

```
SnpMatrixToCount(matList, pedList)
```

### Arguments

| | |
|---|---|
| matList | list of SnpMatrices |
| pedList | list of pedigrees |

### Value

matrix in LINKAGE format

### Examples

```
data(samplePedigrees)
data(snpMat)
ped <- samplePedigrees$secondCousinTriple
ex.ped.mat <- SnpMatrixToCount(list(snpMat), list(ped))
```

## twoFounderSharingProb — *sharing probability when founder pair introduces variant*

### Description

In the case of relatedness among founders, assume that up to two founders could introduce the variant and condition on all possible pairs.

### Usage

```
twoFounderSharingProb(procPed, kinshipCoeff, kinshipOrder)
```

### Arguments

| | |
|---|---|
| procPed | pedigree that has been through processPedigree() |
| kinshipCoeff | mean kinship coefficient among the founders |
| kinshipOrder | order of the polynomial approximation to the distribtion of the number of distinct alleles in the founders (d in Bureau et al.). Must be <= 5 |

### Value

sharing probability

# Index