

# Package ‘GENESIS’

October 13, 2019

**Type** Package

**Title** GENetic ESTimation and Inference in Structured samples  
(GENESIS): Statistical methods for analyzing genetic data from  
samples with population structure and/or relatedness

**Version** 2.15.7

**Date** 2019-10-10

**Author** Matthew P. Conomos, Stephanie M. Gogarten,  
Lisa Brown, Han Chen, Thomas Lumley, Ken Rice, Tamar Sofer, Timothy Thornton, Chaoyu Yu

**Maintainer** Stephanie M. Gogarten <sdmorris@uw.edu>

**Description** The GENESIS package provides methodology for estimating, inferring, and accounting for population and pedigree structure in genetic analyses. The current implementation provides functions to perform PC-AiR (Conomos et al., 2015, Gen Epi) and PC-Relate (Conomos et al., 2016, AJHG). PC-AiR performs a Principal Components Analysis on genome-wide SNP data for the detection of population structure in a sample that may contain known or cryptic relatedness. Unlike standard PCA, PC-AiR accounts for relatedness in the sample to provide accurate ancestry inference that is not confounded by family structure. PC-Relate uses ancestry representative principal components to adjust for population structure/ancestry and accurately estimate measures of recent genetic relatedness such as kinship coefficients, IBD sharing probabilities, and inbreeding coefficients. Additionally, functions are provided to perform efficient variance component estimation and mixed model association testing for both quantitative and binary phenotypes.

**License** GPL-3

**URL** <https://github.com/UW-GAC/GENESIS>

## Depends

**Imports** Biobase, BiocGenerics, GWASTools, gdsfmt, GenomicRanges, IRanges, S4Vectors, SeqArray, SeqVarTools, SNPRelate, data.table, dplyr, foreach, graphics, grDevices, igraph, Matrix, methods, reshape2, stats, utils

**Suggests** CompQuadForm, poibin, survey, testthat, BiocStyle, knitr, rmarkdown, GWASdata, ggplot2, GGally, RColorBrewer, TxDb.Hsapiens.UCSC.hg19.knownGene

**VignetteBuilder** knitr

**biocViews** SNP, GeneticVariability, Genetics, StatisticalMethod,  
DimensionReduction, PrincipalComponent, GenomeWideAssociation,  
QualityControl, BiocViews

**NeedsCompilation** no

**git\_url** <https://git.bioconductor.org/packages/GENESIS>

**git\_branch** master

**git\_last\_commit** 6a317e6

**git\_last\_commit\_date** 2019-10-10

**Date/Publication** 2019-10-12

## R topics documented:

|                                  |    |
|----------------------------------|----|
| GENESIS-package . . . . .        | 2  |
| admixmap . . . . .               | 3  |
| assocTestAggregate . . . . .     | 6  |
| assocTestSingle . . . . .        | 11 |
| fitNullModel . . . . .           | 15 |
| GENESIS-defunct . . . . .        | 19 |
| HapMap_ASW_MXL_KINGmat . . . . . | 20 |
| kin2gds . . . . .                | 20 |
| kingToMatrix . . . . .           | 21 |
| makeSparseMatrix . . . . .       | 23 |
| pcair . . . . .                  | 24 |
| pcairPartition . . . . .         | 27 |
| pccrelate . . . . .              | 29 |
| pccrelateToMatrix . . . . .      | 33 |
| plot.pcair . . . . .             | 34 |
| print.pcair . . . . .            | 36 |
| sample_annotation_1KG . . . . .  | 37 |
| varCompCI . . . . .              | 37 |

**Index** **40**

---

|                 |   |
|-----------------|---|
| GENESIS-package | <i>GENetic Estimation and Inference in Structured samples (GENESIS):<br/>Statistical methods for analyzing genetic data from samples with pop-<br/>ulation structure and/or relatedness</i> |
|-----------------|---|

---

## Description

The GENESIS package provides methodology for estimating, inferring, and accounting for population and pedigree structure in genetic analyses. The current implementation performs PC-AiR (Conomos et al., 2015, Gen Epi) and PC-Relate (Conomos et al., 2016, AJHG). PC-AiR performs a Principal Components Analysis on genome-wide SNP data for the detection of population structure in a sample that may contain known or cryptic relatedness. Unlike standard PCA, PC-AiR accounts for relatedness in the sample to provide accurate ancestry inference that is not confounded by family structure. PC-Relate uses ancestry representative principal components to adjust for population structure/ancestry and accurately estimate measures of recent genetic relatedness such as kinship coefficients, IBD sharing probabilities, and inbreeding coefficients. Additionally, functions are provided to perform efficient variance component estimation and mixed model association testing for both quantitative and binary phenotypes.

## Details

The PC-AiR analysis is performed using the `pcair` function, which takes genotype data and pairwise measures of kinship and ancestry divergence as input and returns PC-AiR PCs as the output. The function `pcairPartition` is called within `pcair` and uses the PC-AiR algorithm to partition the sample into an ancestry representative ‘unrelated subset’ and ‘related subset’. The function `plot.pcair` can be used to plot pairs of PCs from a class ‘pcair’ object returned by the function `pcair`. The function `kingToMatrix` can be used to convert output text files from the KING software (Manichaikul et al., 2010) into an R matrix of pairwise kinship coefficient estimates in a format that can be used by the functions `pcair` and `pcairPartition`. The PC-Relate analysis is performed using the `pcrelate` function, which takes genotype data and PCs from PC-AiR and returns estimates of kinship coefficients, IBD sharing probabilities, and inbreeding coefficients. There are two functions required to perform SNP genotype association testing with mixed models. First, `fitNullModel` is called to fit the null model (i.e. no SNP genotype term) including fixed effects covariates, such as PC-AiR PCs, and random effects specified by their covariance structures, such as a kinship matrix created from PC-Relate output using `pcrelateToMatrix`. The function `fitNullModel` uses AIREML to estimate variance components for the random effects, and the function `varCompCI` can be used to find confidence intervals on the estimates as well as the proportion of total variability they explain; this allows for heritability estimation. Second, `assocTestSingle` is called with the null model output and the genotype data to perform either Wald or score based association tests.

## Author(s)

Matthew P. Conomos, Stephanie M. Gogarten, Lisa Brown, Han Chen, Ken Rice, Tamar Sofer, Timothy Thornton, Chaoyu Yu

Maintainer: Stephanie M. Gogarten <sdmorris@uw.edu>

## References

Conomos M.P., Miller M., & Thornton T. (2015). Robust Inference of Population Structure for Ancestry Prediction and Correction of Stratification in the Presence of Relatedness. *Genetic Epidemiology*, 39(4), 276-293.

Conomos M.P., Reiner A.P., Weir B.S., & Thornton T.A. (2016). Model-free Estimation of Recent Genetic Relatedness. *American Journal of Human Genetics*, 98(1), 127-148.

Manichaikul, A., Mychaleckyj, J.C., Rich, S.S., Daly, K., Sale, M., & Chen, W.M. (2010). Robust relationship inference in genome-wide association studies. *Bioinformatics*, 26(22), 2867-2873.

---

admixMap

*admixMap*

---

## Description

Run admixture analyses

## Usage

```
admixMap(admixDataList, null.model, male.diploid=TRUE,
         genome.build=c("hg19", "hg38"), verbose=TRUE)
```

**Arguments**

|                            |   |
|----------------------------|---|
| <code>admixDataList</code> | named list of <a href="#">GenotypeIterator</a> or <a href="#">SeqVarIterator</a> objects for each ancestry      |
| <code>null.model</code>    | A null model object returned by <a href="#">fitNullModel</a> .  |
| <code>male.diploid</code>  | Logical for whether males on sex chromosomes are coded as diploid.  |
| <code>genome.build</code>  | A character sting indicating genome build; used to identify pseudoautosomal regions on the X and Y chromosomes. |
| <code>verbose</code>       | Logical indicator of whether updates from the function should be printed to the console; the default is TRUE.   |

**Details**

This function is used with local ancestry results such as those obtained from RFMix. RFMix output may be converted to PLINK format, and then to GDS with [snpgdsBED2GDS](#).

`admixDataList` should have one value for each ancestry to be included in the test. The sum of all local ancestries at a particular locus must add up to 2, so if there are K ancestry groups, then only K-1 genotypes can be included since one local ancestry count can be written as a linear combination of all of the other local ancestry counts, resulting in collinearity and a matrix that won't be invertible.

See the example for how one might set up the `admixDataList` object. List names will propagate to the output file.

**Value**

A data.frame where each row refers to a different variant with the columns:

|                         |   |
|-------------------------|---|
| <code>variant.id</code> | The variant ID  |
| <code>chr</code>        | The chromosome value  |
| <code>pos</code>        | The base pair position  |
| <code>n.obs</code>      | The number of samples with non-missing genotypes  |
| <code>*.freq</code>     | The estimated frequency of alleles derived from each ancestry at that variant   |
| <code>*.Est</code>      | The effect size estimate for each additional copy of an allele derived from each ancestry, relative to the reference ancestry |
| <code>*.SE</code>       | The estimated standard error of the effect size estimate for each ancestry  |
| <code>Joint.Stat</code> | The chi-square Wald test statistic for the joint test of all local ancestry terms   |
| <code>Joint.pval</code> | The Wald p-value for the joint test of all local ancestry terms   |

**Author(s)**

Matthew P. Conomos, Lisa Brown, Stephanie M. Gogarten, Tamar Sofer, Ken Rice, Chaoyu Yu

**References**

- Brown, L.A. et al. (2017). Admixture Mapping Identifies an Amerindian Ancestry Locus Associated with Albuminuria in Hispanics in the United States. *J Am Soc Nephrol.* 28(7):2211-2220.
- Maples, B.K. et al. (2013). RFMix: a discriminative modeling approach for rapid and robust local-ancestry inference. *Am J Hum Genet.* 93(2):278-88.

**See Also**

[GenotypeIterator](#), [fitNullModel](#), [assocTestSingle](#)

**Examples**

```

library(GWASTools)

# option 1: one GDS file per ancestry
afrfile <- system.file("extdata", "HapMap_ASW_MXL_local_afr.gds", package="GENESIS")
amerfile <- system.file("extdata", "HapMap_ASW_MXL_local_amer.gds", package="GENESIS")
eurfile <- system.file("extdata", "HapMap_ASW_MXL_local_eur.gds", package="GENESIS")
files <- list(afr=afrfile, amer=amerfile, eur=eurfile)
gdsList <- lapply(files, GdsGenotypeReader)

# make ScanAnnotationDataFrame
scanAnnot <- ScanAnnotationDataFrame(data.frame(
  scanID=getScanID(gdsList[[1]]), stringsAsFactors=FALSE))

# generate a phenotype
set.seed(4)
nsamp <- nrow(scanAnnot)
scanAnnot$pheno <- rnorm(nsamp, mean=0, sd=1)
set.seed(5)
scanAnnot$covar <- sample(0:1, nsamp, replace=TRUE)

genoDataList <- lapply(gdsList, GenotypeData, scanAnnot=scanAnnot)

# iterators
# if we have 3 ancestries total, only 2 should be included in test
genoIterators <- lapply(genoDataList[1:2], GenotypeBlockIterator)

# fit the null mixed model
null.model <- fitNullModel(scanAnnot, outcome="pheno", covars="covar")

# run the association test
myassoc <- admixMap(genoIterators, null.model)
head(myassoc)

lapply(genoDataList, close)

# option 2: create a single file with multiple ancestries
# first, get dosages from all ancestries
library(gdsfmt)
dosages <- lapply(files, function(f) {
  gds <- openfn.gds(f)
  geno <- read.gdsn(index.gdsn(gds, "genotype"))
  closefn.gds(gds)
  geno
})
lapply(dosages, dim)

# create a new file with three dosage matrices, keeping all
# sample and snp nodes from one original file
tmpfile <- tempfile()
file.copy(afrfile, tmpfile)
gds <- openfn.gds(tmpfile, readonly=FALSE)
delete.gdsn(index.gdsn(gds, "genotype"))
add.gdsn(gds, "dosage_afr", dosages[["afr"]])
add.gdsn(gds, "dosage_amer", dosages[["amer"]])

```

```

add.gdsn(gds, "dosage_eur", dosages[["eur"]])
closefn.gds(gds)
cleanup.gds(tmpfile)

# read in GDS data, specifying the node for each ancestry
gds <- openfn.gds(tmpfile)
gds
genoDataList <- list()
for (anc in c("afr", "amer", "eur")){
  gdsr <- GdsGenotypeReader(gds, genotypeVar=paste0("dosage_", anc))
  genoDataList[[anc]] <- GenotypeData(gdsr, scanAnnot=scanAnnot)
}

# iterators
genoIterators <- lapply(genoDataList[1:2], GenotypeBlockIterator)

# run the association test
myassoc <- admixMap(genoIterators, null.model)

close(genoDataList[[1]])
unlink(tmpfile)

```

---

assocTestAggregate      *Aggregate Association Testing*

---

## Description

assocTestAggregate performs aggregate association tests using the null model fit with [fitNullModel](#).

## Usage

```

## S4 method for signature 'SeqVarIterator'
assocTestAggregate(gdsobj, null.model, AF.max=1,
  weight.beta=c(1,1), weight.user=NULL,
  test=c("Burden", "SKAT", "fastSKAT", "SMMAT", "SKATO"),
  burden.test=c("Score", "Wald"),
  neig = 200, ntrace = 500,
  rho = seq(from = 0, to = 1, by = 0.1),
  sparse=TRUE, imputed=FALSE,
  male.diploid=TRUE, genome.build=c("hg19", "hg38"),
  verbose=TRUE)

## S4 method for signature 'GenotypeIterator'
assocTestAggregate(gdsobj, null.model, AF.max=1,
  weight.beta=c(1,1), weight.user=NULL,
  test=c("Burden", "SKAT", "fastSKAT", "SMMAT", "SKATO"),
  burden.test=c("Score", "Wald"),
  neig = 200, ntrace = 500,
  rho = seq(from = 0, to = 1, by = 0.1),
  male.diploid=TRUE, verbose=TRUE)

```

**Arguments**

|                           |   |
|---------------------------|---|
| <code>gdsobj</code>       | An object of class <code>SeqVarIterator</code> from the package <code>SeqVarTools</code> containing the genotype data for the variants and samples to be used for the analysis.   |
| <code>null.model</code>   | A null model object returned by <code>fitNullModel</code> .   |
| <code>AF.max</code>       | A numeric value specifying the upper bound on the alternate allele frequency for variants to be included in the analysis.   |
| <code>weight.beta</code>  | A numeric vector of length two specifying the two parameters of the Beta distribution used to determine variant weights; weights are given by <code>dbeta(MAF, a, b)</code> , where MAF is the minor allele frequency, and a and b are the two parameters specified here. <code>weight.beta = c(1, 25)</code> gives the Wu weights; <code>weight.beta = c(0.5, 0.5)</code> is proportional to the Madsen-Browning weights; and <code>weight.beta = c(1, 1)</code> gives a weight of 1 to all variants. This input is ignored when <code>weight.user</code> is not NULL. |
| <code>weight.user</code>  | A character string specifying the name of a variable to be used as variant weights. This variable can be in either 1) the <code>variantData</code> slot of <code>gdsobj</code> or 2) the <code>mcols</code> of the <code>GRanges</code> or <code>GRangesList</code> object used to create <code>gdsobj</code> (when <code>gdsobj</code> is a <code>link{SeqVarRangeIterator}</code> or <code>link{SeqVarListIterator}</code> ). When left NULL (the default), the weights specified by <code>weight.beta</code> will be used.   |
| <code>test</code>         | A character string specifying the type of test to be performed. The possibilities are "Burden" (default), "SKAT", "fastSKAT", "SMMAT", or "SKATO".  |
| <code>burden.test</code>  | A character string specifying the type of Burden test to perform when <code>test = "Burden"</code> . The possibilities are "Score" and "Wald". "Score" can be used for any <code>null.model</code> . "Wald" can not be used when the <code>null.model</code> is from a mixed model with a binary outcome variable.  |
| <code>neig</code>         | The number eigenvalues to approximate by using random projections for calculating p-values with fastSKAT; default is 200. See 'Details' for more information.   |
| <code>ntrace</code>       | The number of vectors to sample when using random projections to estimate the trace needed for p-value calculation with fastSKAT; default is 500. See 'Details' for more information.   |
| <code>rho</code>          | A numeric value (or vector of numeric values) in $[0, 1]$ specifying the rho parameter when using <code>test == "SKATO"</code> ; these are the values for which SKAT-O is performed, defining the search space for the optimal rho. If <code>rho = 0</code> , this is equivalent to a standard SKAT test; if <code>rho = 1</code> , this is equivalent to a score burden test.  |
| <code>sparse</code>       | Logical indicator of whether to read genotypes as sparse Matrix objects; the default is TRUE. Set this to FALSE if the alternate allele dosage of the genotypes in the test are not expected to be mostly 0.  |
| <code>imputed</code>      | Logical indicator of whether to read dosages from the "DS" field containing imputed dosages instead of counting the number of alternate alleles.  |
| <code>male.diploid</code> | Logical for whether males on sex chromosomes are coded as diploid.  |
| <code>genome.build</code> | A character sting indicating genome build; used to identify pseudoautosomal regions on the X and Y chromosomes.   |
| <code>verbose</code>      | Logical indicator of whether updates from the function should be printed to the console; the default is TRUE.   |

## Details

The type of aggregate unit tested depends on the class of iterator used for `gdsobj`. Options include sliding windows, specific ranges of variants or selection of individual variants (ranges with width 1). See [SeqVarIterator](#) for more details.

The effect size estimate is for each copy of the alternate allele. For multiallelic variants, each alternate allele is tested separately.

Monomorphic variants (including variants where every sample is a heterozygote) are always omitted from the aggregate unit prior to testing.

Somewhat similarly to SKAT-O, the variant Set Mixed Model Association Test (SMMAT, Chen et al., 2019) combines the burden test p-value with an adjusted SKAT (which is asymptotically independent of the burden test) p-value using a chi-square distribution with 4df from Fisher's method.

SKAT and SMMAT will attempt to use Davies' method (i.e. integration) to calculate p-values; if an error occurs in integration or the reported p-values are too small that they are unreliable (i.e. near machine epsilon), then the saddlepoint approximation will instead be used to calculate the p-values.

The fastSKAT method of Lumley et al. (2018) uses random matrix theory to speed up the computation of SKAT p-values. When `test = "fastSKAT"`, the function attempts to intelligently determine which p-value calculation approach to use for each aggregation unit: (1) if `min(number samples, number variants)` is small enough, then the standard SKAT p-value calculation is used; (2) if `min(number samples, number variants)` is too large for standard SKAT, but small enough to explicitly compute the genotype covariance matrix, random projections are used to approximate the eigenvalues of the covariance matrix, and the fastSKAT p-value calculation is used; (3) if `min(number samples, number variants)` is too big to explicitly compute the genotype covariance matrix, random projections are used to approximate both the eigenvalues and the trace of the covariance matrix, and the fastSKAT p-value calculation is used.)

## Value

A list with the following items:

`results` A data.frame containing the results from the main analysis. Each row is a separate aggregate test:

If `gdsobj` is a [SeqVarWindowIterator](#):

`chr` The chromosome value  
`start` The start position of the window  
`end` The end position of the window

Always:

`n.site` The number of variant sites included in the test.  
`n.alt` The number of alternate alleles included in the test.  
`n.sample.alt` The number of samples with an observed alternate allele at any variant in the aggregate set.

If `test` is "Burden":

If `burden.test` is "Score":

`Score` The value of the score function  
`Score.SE` The estimated standard error of the Score  
`Score.Stat` The score Z test statistic



|                                  |  |
|----------------------------------|--|
| Score.pval                       | The score p-value  |
| If burden.test is "Wald":        |  |
| Est                              | The effect size estimate for a one unit increase in the burden value   |
| Est.SE                           | The estimated standard error of the effect size estimate   |
| Wald.Stat                        | The Wald Z test statistic  |
| Wald.pval                        | The Wald p-value   |
| If test is "SKAT" or "fastSKAT": |  |
| Q                                | The SKAT test statistic.   |
| pval                             | The SKAT p-value.  |
| err                              | Takes value 1 if there was an error in calculating the p-value; takes the value 2 when multiple random projections were required to get a good approximation from fastSKAT (the reported p-value is likely still reliable); 0 otherwise.   |
| pval.method                      | The p-value calculation method used. When standard SKAT is used, one of "integration" or "saddlepoint"; when fastSKAT random projections are used to approximate eigenvalues of the genotype covariance matrix, one of "ssvd_integration" or "ssvd_saddlepoint"; when fastSKAT random projections are used to approximate both the eigenvalues and the trace of the genotype covariance matrix, one of "rsvd_integration" or "rsvd_saddlepoint". |
| If test is "SMMAT":              |  |
| pval_burden                      | The burden test p-value  |
| pval_theta                       | The p-value of the adjusted SKAT test (which is asymptotically independent of the burden test)   |
| pval_SMMAT                       | The SMMAT p-value after combining pval_burden and pval_theta using Fisher's method.  |
| err                              | Takes value 1 if there was an error calculating the SMMAT p-value; 0 otherwise. If err=1, pval_SMMAT is set to pval_burden.  |
| If test is "SKATO":              |  |
| Q_rho                            | The SKAT test statistic for the value of rho specified. There will be as many of these variables as there are rho values chosen.   |
| pval_rho                         | The SKAT p-value for the value of rho specified. There will be as many of these variables as there are rho values chosen.  |
| err_rho                          | Takes value 1 if there was an error in calculating the p-value for the value of rho specified when using the "kuonen" or "davies" methods; 0 otherwise. When there is an error, the p-value returned is from the "liu" method. There will be as many of these variables as there are rho values chosen.  |
| min.pval                         | The minimum p-value among the p-values calculated for each choice of rho.  |
| opt_rho                          | The optimal rho value; i.e. the rho value that gave the minimum p-value.   |
| pval_SKATO                       | The SKAT-O p-value after adjustment for searching across multiple rho values.  |
| variantInfo                      | A list with as many elements as aggregate tests performed. Each element of the list is a data.frame providing information on the variants used in the aggregate test with results presented in the corresponding row of results. Each of these data.frames has the following information:  |

|              |   |
|--------------|---|
| variant.id   | The variant ID  |
| chr          | The chromosome value  |
| pos          | The base pair position  |
| allele.index | The index of the alternate allele. For biallelic variants, this will always be 1.   |
| n.obs        | The number of samples with non-missing genotypes  |
| freq         | The estimated alternate allele frequency  |
| MAC          | The minor allele count. For multiallelic variants, "minor" is determined by comparing the count of the alternate allele specified by <code>allele.index</code> with the sum of all other alleles. |
| weight       | The weight assigned to the variant in the analysis.   |

### Author(s)

Matthew P. Conomos, Stephanie M. Gogarten, Thomas Lumley, Tamar Sofer, Ken Rice, Chaoyu Yu, Han Chen

### References

- Leal, S.M. & Li, B. (2008). Methods for Detecting Associations with Rare Variants for Common Diseases: Application to Analysis of Sequence Data. *American Journal of Human Genetics*, 83(3): 311-321.
- Browning, S.R. & Madsen, B.E. (2009). A Groupwise Association Test for Rare Mutations Using a Weighted Sum Statistic. *PLoS Genetics*, 5(2): e1000384.
- Wu, M.C, Lee, S., Cai, T., Li, Y., Boehnke, M., & Lin, X. (2011). Rare-Variant Association Testing for Sequencing Data with the Sequence Kernel Association Test. *American Journal of Human Genetics*, 89(1): 82-93.
- Lee, S. et al. (2012). Optimal Unified Approach for Rare-Variant Association Testing with Application to Small-Sample Case-Control Whole-Exome Sequencing Studies. *American Journal of Human Genetics*, 91(2): 224-237.
- Chen, H., Huffman, J. E., Brody, J. A., Wang, C., Lee, S., Li, Z., ... & Blangero, J. (2019). Efficient variant set mixed model association tests for continuous and binary traits in large-scale whole-genome sequencing studies. *The American Journal of Human Genetics*, 104(2), 260-274.
- Lumley, T., Brody, J., Peloso, G., Morrison, A., & Rice, K. (2018). FastSKAT: Sequence kernel association tests for very large sets of markers. *Genetic epidemiology*, 42(6), 516-527.

### Examples

```
library(SeqVarTools)
library(Biobase)
library(GenomicRanges)

# open a sequencing GDS file
gdsfile <- seqExampleFileName("gds")
gds <- seqOpen(gdsfile)

# simulate some phenotype data
set.seed(4)
data(pedigree)
pedigree <- pedigree[match(seqGetData(gds, "sample.id"), pedigree$sample.id),]
pedigree$outcome <- rnorm(nrow(pedigree))
```

```

# construct a SeqVarData object
seqData <- SeqVarData(gds, sampleData=AnnotatedDataFrame(pedigree))

# fit the null model
nullmod <- fitNullModel(seqData, outcome="outcome", covars="sex")

# burden test - Range Iterator
gr <- GRanges(seqnames=rep(1,3), ranges=IRanges(start=c(1e6, 2e6, 3e6), width=1e6))
iterator <- SeqVarRangeIterator(seqData, variantRanges=gr)
assoc <- assocTestAggregate(iterator, nullmod, test="Burden")
assoc$results
lapply(assoc$variantInfo, head)

# SKAT test - Window Iterator
seqSetFilterChrom(seqData, include="22")
iterator <- SeqVarWindowIterator(seqData)
assoc <- assocTestAggregate(iterator, nullmod, test="SKAT")
head(assoc$results)
head(assoc$variantInfo)

# SKAT-O test - List Iterator
seqResetFilter(iterator)
gr <- GRangesList(
  GRanges(seqnames=rep(22,2), ranges=IRanges(start=c(16e6, 17e6), width=1e6)),
  GRanges(seqnames=rep(22,2), ranges=IRanges(start=c(18e6, 20e6), width=1e6)))
iterator <- SeqVarListIterator(seqData, variantRanges=gr)
assoc <- assocTestAggregate(iterator, nullmod, test="SKAT", rho=seq(0, 1, 0.25))
assoc$results
assoc$variantInfo

# user-specified weights - option 1
seqResetFilter(iterator)
variant.id <- seqGetData(gds, "variant.id")
weights <- data.frame(variant.id, weight=runif(length(variant.id)))
variantData(seqData) <- AnnotatedDataFrame(weights)
iterator <- SeqVarListIterator(seqData, variantRanges=gr)
assoc <- assocTestAggregate(iterator, nullmod, test="Burden", weight.user="weight")
assoc$results
assoc$variantInfo

# user-specified weights - option 2
seqResetFilter(iterator)
variantData(seqData)$weight <- NULL
gr <- GRangesList(
  GRanges(seqnames=rep(22,2), ranges=IRanges(start=c(16e6, 17e6), width=1e6), weight=runif(2)),
  GRanges(seqnames=rep(22,2), ranges=IRanges(start=c(18e6, 20e6), width=1e6), weight=runif(2)))
iterator <- SeqVarListIterator(seqData, variantRanges=gr)
assoc <- assocTestAggregate(iterator, nullmod, test="Burden", weight.user="weight")
assoc$results
assoc$variantInfo

seqClose(seqData)

```

## Description

assocTestSingle performs genotype association tests using the null model fit with [fitNullModel](#).

## Usage

```
## S4 method for signature 'SeqVarIterator'
assocTestSingle(gdsobj, null.model, test=c("Score", "Wald"),
                GxE=NULL, sparse=TRUE, imputed=FALSE, male.diploid=TRUE,
                genome.build=c("hg19", "hg38"), verbose=TRUE)
## S4 method for signature 'GenotypeIterator'
assocTestSingle(gdsobj, null.model, test=c("Score", "Wald"),
                GxE=NULL, male.diploid=TRUE, verbose=TRUE)
```

## Arguments

|              |   |
|--------------|---|
| gdsobj       | An object of class <a href="#">SeqVarIterator</a> from the package <a href="#">SeqVarTools</a> , or an object of class <a href="#">GenotypeIterator</a> from the package <a href="#">GWASTools</a> , containing the genotype data for the variants and samples to be used for the analysis. |
| null.model   | A null model object returned by <a href="#">fitNullModel</a> .  |
| test         | A character string specifying the type of test to be performed. The possibilities are "Score" (default) or "Wald"; only "Score" can be used when the family of the null model fit with <a href="#">fitNullModel</a> is not gaussian.  |
| GxE          | A vector of character strings specifying the names of the variables for which a genotype interaction term should be included. If NULL (default) no genotype interactions are included. See 'Details' for further information.   |
| sparse       | Logical indicator of whether to read genotypes as sparse Matrix objects; the default is TRUE. Set this to FALSE if the alternate allele dosage of the genotypes in the test are not expected to be mostly 0.  |
| imputed      | Logical indicator of whether to read dosages from the "DS" field containing imputed dosages instead of counting the number of alternate alleles.  |
| male.diploid | Logical for whether males on sex chromosomes are coded as diploid.  |
| genome.build | A character sting indicating genome build; used to identify pseudoautosomal regions on the X and Y chromosomes.   |
| verbose      | Logical indicator of whether updates from the function should be printed to the console; the default is TRUE.   |

## Details

The effect size estimate is for each copy of the alternate allele. For multiallelic variants, each alternate allele is tested separately.

Sporadic missing genotype values are mean imputed using the minor allele frequency (MAF) calculated on all other samples at that variant.

Monomorphic variants (including variants where every sample is a heterozygote) are omitted from the results.

The input GxE can be used to perform GxE tests. Multiple interaction variables may be specified, but all interaction variables specified must have been included as covariates in fitting the null model with [fitNullModel](#). When performing GxE analyses, [assocTestSingle](#) will report two tests: (1) the joint test of all genotype interaction terms in the model (this is the test for any genotype interaction effect), and (2) the joint test of the genotype term along with all of the genotype interaction terms

(this is the test for any genetic effect). Individual genotype interaction terms can be tested by creating Wald test statistics from the reported effect size estimates and their standard errors (Note: when GxE contains a single continuous or binary covariate, this test is the same as the test for any genotype interaction effect mentioned above).

For the [GenotypeIterator](#) method, objects created with [GdsGenotypeReader](#) or [MatrixGenotypeReader](#) are supported. [NcdfGenotypeReader](#) objects are not supported.

## Value

A data.frame where each row refers to a different variant with the columns:

|              |   |
|--------------|---|
| variant.id   | The variant ID  |
| chr          | The chromosome value  |
| pos          | The base pair position  |
| allele.index | The index of the alternate allele. For biallelic variants, this will always be 1.   |
| n.obs        | The number of samples with non-missing genotypes  |
| freq         | The estimated alternate allele frequency  |
| MAC          | The minor allele count. For multiallelic variants, "minor" is determined by comparing the count of the alternate allele specified by <code>allele.index</code> with the sum of all other alleles. |

If test is "Score":

|            |   |
|------------|---|
| Score      | The value of the score function           |
| Score.SE   | The estimated standard error of the Score |
| Score.Stat | The score Z test statistic                |
| Score.pval | The score p-value                         |

If test is "Wald" and GxE is NULL:

|           |   |
|-----------|---|
| Est       | The effect size estimate for each additional copy of the alternate allele |
| Est.SE    | The estimated standard error of the effect size estimate                  |
| Wald.Stat | The Wald Z test statistic   |
| Wald.pval | The Wald p-value  |

If test is "Wald" and GxE is not NULL:

|            |  |
|------------|--|
| Est.G      | The effect size estimate for the genotype term   |
| Est.G:env  | The effect size estimate for the genotype*env interaction term. There will be as many of these terms as there are interaction variables, and "env" will be replaced with the variable name.        |
| SE.G       | The estimated standard error of the genotype term effect size estimate   |
| SE.G:env   | The estimated standard error of the genotype*env effect size estimate. There will be as many of these terms as there are interaction variables, and "env" will be replaced with the variable name. |
| GxE.Stat   | The Wald Z test statistic for the test of all genotype interaction terms. When there is only one genotype interaction term, this is the test statistic for that term.                              |
| GxE.pval   | The Wald p-value for the test of all genotype interaction terms; i.e. the test of any genotype interaction effect  |
| Joint.Stat | The Wald Z test statistic for the joint test of the genotype term and all of the genotype interaction terms  |
| Joint.pval | The Wald p-value for the joint test of the genotype term and all of the genotype interaction terms; i.e. the test of any genotype effect   |

**Author(s)**

Matthew P. Conomos, Stephanie M. Gogarten, Tamar Sofer, Ken Rice, Chaoyu Yu

**See Also**

[fitNullModel](#) for fitting the null mixed model needed as input to `assocTestSingle`. [SeqVarIterator](#) for creating the input object with genotypes.

**Examples**

```
library(SeqVarTools)
library(Biobase)

# open a sequencing GDS file
gdsfile <- seqExampleFileName("gds")
gds <- seqOpen(gdsfile)

# simulate some phenotype data
set.seed(4)
data(pedigree)
pedigree <- pedigree[match(seqGetData(gds, "sample.id"), pedigree$sample.id),]
pedigree$outcome <- rnorm(nrow(pedigree))

# construct a SeqVarIterator object
seqData <- SeqVarData(gds, sampleData=AnnotatedDataFrame(pedigree))
iterator <- SeqVarBlockIterator(seqData)

# fit the null model
nullmod <- fitNullModel(iterator, outcome="outcome", covars="sex")

# run the association test
assoc <- assocTestSingle(iterator, nullmod)

seqClose(iterator)

library(GWASTools)

# open a SNP-based GDS file
gdsfile <- system.file("extdata", "HapMap_ASW_MXL_geno.gds", package="GENESIS")
gds <- GdsGenotypeReader(filename = gdsfile)

# simulate some phenotype data
set.seed(4)
pheno <- data.frame(scanID=getScanID(gds),
                    outcome=rnorm(nscan(gds)))

# construct a GenotypeIterator object
genoData <- GenotypeData(gds, scanAnnot=ScanAnnotationDataFrame(pheno))
iterator <- GenotypeBlockIterator(genoData)

# fit the null model
nullmod <- fitNullModel(iterator, outcome="outcome")

# run the association test
assoc <- assocTestSingle(iterator, nullmod)
```

```
close(iterator)
```

---

```
fitNullModel
```

```
Fit a Model Under the Null Hypothesis
```

---

## Description

`fitNullModel` fits a regression model or a mixed model with random effects specified by their covariance structures; this allows for the inclusion of a polygenic random effect using a kinship matrix or genetic relationship matrix (GRM). The output of `fitNullModel` can be used to estimate genetic heritability and can be passed to `assocTestSingle` or `assocTestAggregate` for the purpose of genetic association testing.

`nullModelInvNorm` does an inverse normal transform of a previously fit null model.

## Usage

```
## S4 method for signature 'data.frame'
fitNullModel(x, outcome, covars = NULL, cov.mat = NULL,
             group.var = NULL, family = "gaussian", start = NULL,
             AIREML.tol = 1e-4, max.iter = 100, EM.iter = 0, drop.zeros = TRUE,
             verbose = TRUE)
## S4 method for signature 'AnnotatedDataFrame'
fitNullModel(x, outcome, covars = NULL, cov.mat = NULL,
             group.var = NULL, sample.id = NULL, ...)
## S4 method for signature 'SeqVarData'
fitNullModel(x, ...)
## S4 method for signature 'ScanAnnotationDataFrame'
fitNullModel(x, ...)
## S4 method for signature 'GenotypeData'
fitNullModel(x, ...)

nullModelInvNorm(null.model, cov.mat = NULL, norm.option = c("by.group", "all"),
                 rescale = c("none", "model", "residSD"),
                 AIREML.tol = 1e-4, max.iter = 100, EM.iter = 0, verbose = TRUE)
```

## Arguments

|                        |   |
|------------------------|---|
| <code>x</code>         | An object of class <code>data.frame</code> , <code>AnnotatedDataFrame</code> , or <code>SeqVarData</code> containing the outcome and covariate data for the samples to be used for the analysis.  |
| <code>outcome</code>   | A character string specifying the name of the outcome variable in <code>x</code> .  |
| <code>covars</code>    | A vector of character strings specifying the names of the fixed effect covariates in <code>x</code> ; an intercept term is automatically included. If <code>NULL</code> (default) the only fixed effect covariate is the intercept term.  |
| <code>cov.mat</code>   | A matrix or list of matrices specifying the covariance structures of the random effects terms. Objects from the <code>Matrix</code> package are supported. See 'Details' for more information.  |
| <code>group.var</code> | This variable can only be used when <code>family = "gaussian"</code> . A character string specifying the name of a categorical variable in <code>x</code> that is used to fit heterogeneous residual error variances. If <code>NULL</code> (default), then a standard LMM with constant residual variance for all samples is fit. See 'Details' for more information. |

|                          |   |
|--------------------------|---|
| <code>sample.id</code>   | A vector of IDs for samples to include in the analysis. If NULL, all samples in <code>x</code> are included. This argument is ignored if <code>x</code> is a <code>data.frame</code> ; see 'Details'.   |
| <code>family</code>      | A description of the error distribution to be used in the model. The default "gaussian" fits a linear model; see <a href="#">family</a> for further options, and see 'Details' for more information.  |
| <code>start</code>       | A vector of starting values for the variance component estimation procedure. The function will pick reasonable starting values when left NULL (default). See 'Details' for more information.  |
| <code>AIREML.tol</code>  | The convergence threshold for the Average Information REML (AIREML) procedure used to estimate the variance components of the random effects. See 'Details' for more information.   |
| <code>max.iter</code>    | The maximum number of iterations allowed to reach convergence.  |
| <code>EM.iter</code>     | The number of EM iterations to run prior to AIREML; default is 0.   |
| <code>drop.zeros</code>  | Logical indicator of whether variance component terms that converge to 0 should be removed from the model; the default is TRUE. See 'Details' for more information.   |
| <code>verbose</code>     | Logical indicator of whether updates from the function should be printed to the console; the default is TRUE.   |
| <code>...</code>         | Arguments to pass to other methods.   |
| <code>null.model</code>  | The output of <code>fitNullModel</code> .   |
| <code>norm.option</code> | Whether the normalization should be done separately within each value of <code>group.var</code> ("by.group") or with all samples together ("all").  |
| <code>rescale</code>     | Controls whether to rescale the variance after inverse-normal transform, restoring it to the original variance before the transform. "none" for no rescaling of the residuals; "model" for model-based rescaling, and "residSD" to rescale to the standard deviation of the marginal residuals. See 'Details' for more information. |

## Details

If `x` is a `data.frame`, the `rownames` of `x` must match the row and column names of `cov.mat` (if `cov.mat` is specified). If `x` is an [AnnotatedDataFrame](#) or other object containing an [AnnotatedDataFrame](#), `x` will be re-ordered (if necessary) so that `sample.id` or `scanID` is in the same order as the row and column names of `cov.mat`.

`cov.mat` is used to specify the covariance structures of the random effects terms in the model. For example, to include a polygenic random effect, one matrix in `cov.mat` could be a kinship matrix or a genetic relationship matrix (GRM). As another example, to include household membership as a random effect, one matrix in `cov.mat` should be a 0/1 matrix with a 1 in the  $[i, j]$  and  $[j, i]$  entries if individuals  $i$  and  $j$  are in the same household and 0 otherwise; the diagonals of such a matrix should all be 1.

When `family` is not gaussian, the penalized quasi-likelihood (PQL) approximation to the generalized linear mixed model (GLMM) is fit following the procedure of GMMAT (Chen et al.).

For some outcomes, there may be evidence that different groups of observations have different residual variances, and the standard LMM assumption of homoscedasticity is violated. When `group.var` is specified, separate (heterogeneous) residual variance components are fit for each unique value of `group.var`.

Let  $m$  be the number of matrices in `cov.mat` and let  $g$  be the number of categories in the variable specified by `group.var`. The length of the `start` vector must be  $(m + 1)$  when `family` is gaussian



and `group.var` is NULL;  $(m + g)$  when `family` is gaussian and `group.var` is specified; or  $m$  when `family` is not gaussian.

A Newton-Raphson iterative procedure with Average Information REML (AIREML) is used to estimate the variance components of the random effects. When the absolute change between all of the new and previous variance component estimates is less than  $\text{var}(\text{outcome}) * \text{AIREML.tol}$ , the algorithm declares convergence of the estimates. Sometimes a variance component may approach the boundary of the parameter space at 0; step-halving is used to prevent any component from becoming negative. However, when a variance component gets near the 0 boundary, the algorithm can sometimes get "stuck", preventing the other variance components from converging; if `drop.zeros` is TRUE, then variance components that converge to a value less than `AIREML.tol` will be dropped from the model and the estimation procedure will continue with the remaining variance components.

After inverse-normal transformation, the variance rescaling is done with the same grouping; i.e. if `norm.option == "by.group"`, rescaling is done within each group, and if `norm.option == "all"`, rescaling is done with all samples.

## Value

An object of class `'GENESIS.nullModel'` or `'GENESIS.nullMixedModel'`. A list including:

|                                |  |
|--------------------------------|--|
| <code>family</code>            | A character string specifying the family used in the analysis.   |
| <code>hetResid</code>          | A logical indicator of whether heterogeneous residual variance components were used in the model (specified by <code>group.var</code> ).   |
| <code>varComp</code>           | The variance component estimates. There is one variance component for each random effect specified in <code>cov.mat</code> . When <code>family</code> is gaussian, there are additional residual variance components; one residual variance component when <code>group.var</code> is NULL, and as many residual variance components as there are unique values of <code>group.var</code> when it is specified. |
| <code>varCompCov</code>        | The estimated covariance matrix of the variance component estimates given by <code>varComp</code> . This can be used for hypothesis tests regarding the variance components.   |
| <code>fixef</code>             | A data.frame with effect size estimates (betas), standard errors, chi-squared test statistics, and p-values for each of the fixed effect covariates specified in <code>covars</code> .   |
| <code>betaCov</code>           | The estimated covariance matrix of the effect size estimates (betas) of the fixed effect covariates. This can be used for hypothesis tests regarding the fixed effects.  |
| <code>fitted.values</code>     | The fitted values from the model; i.e. $W * \text{beta}$ where $W$ is the design matrix and $\text{beta}$ are the effect size estimates for the fixed effects.   |
| <code>resid.marginal</code>    | The marginal residuals from the model; i.e. $Y - W * \text{beta}$ where $Y$ is the vector of outcome values.   |
| <code>resid.conditional</code> | The conditional residuals from the model; i.e. $Y - W * \text{beta} - Z * u$ .   |
| <code>logLik</code>            | The log-likelihood value.  |
| <code>logLikR</code>           | The restricted log-likelihood value.   |
| <code>AIC</code>               | The Akaike Information Criterion value.  |
| <code>workingY</code>          | The "working" outcome vector. When <code>family</code> is gaussian, this is just the original outcome vector. When <code>family</code> is not gaussian, this is the PQL linearization of the outcome vector. This is used by <a href="#">assocTestSingle</a> or <a href="#">assocTestAggregate</a> for genetic association testing. See 'Details' for more information.  |

|              |   |
|--------------|---|
| outcome      | The original outcome vector, as a 1-column matrix with column name. When family is gaussian, this is equal to workingY.   |
| model.matrix | The design matrix for the fixed effect covariates used in the model.  |
| group.idx    | If group.var is not NULL, a list of indices for samples in each group.  |
| cholSigmaInv | The Cholesky decomposition of the inverse of the estimated outcome covariance structure. This is used by <a href="#">assocTestSingle</a> or <a href="#">assocTestAggregate</a> for genetic association testing. |
| converged    | A logical indicator of whether the AIREML procedure for estimating the random effects variance components converged.  |
| zeroFLAG     | A vector of logicals the same length as varComp specifying whether the corresponding variance component estimate was set to 0 by the function due to convergence to the boundary in the AIREML procedure.       |
| RSS          | The residual sum of squares from the model fit. When family is gaussian, this will typically be 1 since the residual variance component is estimated separately.  |
| sample.id    | A vector of IDs for the samples used in the analysis.   |

### Author(s)

Matthew P. Conomos, Stephanie M. Gogarten, Tamar Sofer, Ken Rice, Chaoyu Yu

### References

Chen H, Wang C, Conomos MP, Stilp AM, Li Z, Sofer T, Szpiro AA, Chen W, Brehm JM, Celedon JC, Redline S, Papanicolaou GJ, Thornton TA, Laurie CC, Rice K and Lin X. (2016) Control for Population Structure and Relatedness for Binary Traits in Genetic Association Studies Using Logistic Mixed Models. *American Journal of Human Genetics*, 98(4):653-66.

Breslow NE and Clayton DG. (1993). Approximate Inference in Generalized Linear Mixed Models. *Journal of the American Statistical Association* 88: 9-25.

Gilmour, A.R., Thompson, R., & Cullis, B.R. (1995). Average information REML: an efficient algorithm for variance parameter estimation in linear mixed models. *Biometrics*, 1440-1450.

### See Also

[varCompCI](#) for estimating confidence intervals for the variance components and the proportion of variability (heritability) they explain, [assocTestSingle](#) or [assocTestAggregate](#) for running genetic association tests using the output from `fitNullModel`.

### Examples

```
library(GWASTools)

# file path to GDS file
gdsfile <- system.file("extdata", "HapMap_ASW_MXL_geno.gds", package="GENESIS")
# read in GDS data
HapMap_geno <- GdsGenotypeReader(filename = gdsfile)
# create a GenotypeData class object
HapMap_genoData <- GenotypeData(HapMap_geno)
# load saved matrix of KING-robust estimates
data("HapMap_ASW_MXL_KINGmat")

# run PC-AiR
mypcair <- pcair(HapMap_genoData, kinobj = HapMap_ASW_MXL_KINGmat,
```

```
divobj = HapMap_ASW_MXL_KINGmat)

# run PC-Relate
HapMap_genodata <- GenotypeBlockIterator(HapMap_genodata, snpBlock=20000)
mypcrel <- pcrelate(HapMap_genodata, pcs = mypcair$vectors[,1,drop=FALSE],
  training.set = mypcair$unrels)
close(HapMap_genodata)

# generate a phenotype
set.seed(4)
pheno <- 0.2*mypcair$vectors[,1] + rnorm(mypcair$nsamp, mean = 0, sd = 1)

annot <- data.frame(sample.id = mypcair$sample.id,
  pc1 = mypcair$vectors[,1], pheno = pheno)

# make covariance matrix
cov.mat <- pcrelateToMatrix(mypcrel, verbose=FALSE)[annot$sample.id, annot$sample.id]

# fit the null mixed model
nullmod <- fitNullModel(annot, outcome = "pheno", covars = "pc1", cov.mat = cov.mat)
```

---

GENESIS-defunct

*Defunct functions in package* **GENESIS**

---

## Description

These functions are defunct and no longer available.

## Details

The following functions are defunct; use the replacement indicated below:

- admixMapMM: [admixMap](#)
- assocTestMM: [assocTestSingle](#)
- assocTestSeq: [assocTestAggregate](#)
- assocTestSeqWindow: [assocTestAggregate](#)
- fitNullMM: [fitNullModel](#)
- fitNullReg: [fitNullModel](#)
- king2mat: [kingToMatrix](#)
- pcrelate,GenotypeData-method: [pcrelate,GenotypeIterator-method](#)
- pcrelate,SeqVarData-method: [pcrelate,SeqVarIterator-method](#)
- pcrelateMakeGRM: [pcrelateToMatrix](#)
- pcrelateReadInbreed: [pcrelate](#)
- pcrelateReadKinship: [pcrelate](#)

---

HapMap\_ASW\_MXL\_KINGmat

*Matrix of Pairwise Kinship Coefficient Estimates for the combined HapMap ASW and MXL Sample found with the KING-robust estimator from the KING software.*

---

### Description

KING-robust kinship coefficient estimates for the combined HapMap African Americans in the Southwest U.S. (ASW) and Mexican Americans in Los Angeles (MXL) samples.

### Usage

```
data(HapMap_ASW_MXL_KINGmat)
```

### Format

The format is: num [1:173, 1:173] 0 0.00157 -0.00417 0.00209 0.00172 ...

### Value

A matrix of pairwise kinship coefficient estimates as calculated with KING-robust for the combined HapMap African Americans in the Southwest U.S. (ASW) and Mexican Americans in Los Angeles (MXL) samples.

### Source

<http://hapmap.ncbi.nlm.nih.gov/>

### References

International HapMap 3 Consortium. (2010). Integrating common and rare genetic variation in diverse human populations. *Nature*, 467(7311), 52-58.

---

kin2gds

*Store kinship matrix in GDS*

---

### Description

kin2gds and mat2gds write kinship matrices to GDS files.

### Usage

```
kin2gds(ibdobj, gdsfile)
mat2gds(mat, gdsfile)
```

### Arguments

|         |  |
|---------|--|
| ibdobj  | A list with elements <code>sample.id</code> and <code>kinship</code> , such as the output of <a href="#">snpgdsIBDKING</a> . |
| mat     | A matrix with sample identifiers in colnames.  |
| gdsfile | A character string with the name of the GDS file to create.  |

**Details**

When using `pcair` or `pcairPartition` with large sample sizes, it can be more memory efficient to read data from GDS files. `kin2gds` and `mat2gds` store kinship matrices in GDS files for use with these functions.

**Author(s)**

Stephanie M. Gogarten

**See Also**

[kingToMatrix](#), [snpgdsIBDKING](#)

**Examples**

```
library(gdsfmt)

# KING results from the command-line program
file.kin0 <- system.file("extdata", "MXL_ASW.kin0", package="GENESIS")
file.kin <- system.file("extdata", "MXL_ASW.kin", package="GENESIS")
KINGmat <- kingToMatrix(c(file.kin0, file.kin), estimator="Kinship")
gdsfile <- tempfile()
mat2gds(KINGmat, gdsfile)
gds <- openfn.gds(gdsfile)
gds
closefn.gds(gds)

# KING results from SNPRelate
library(SNPRelate)
geno <- snpgdsOpen(snpgdsExampleFileName())
king <- snpgdsIBDKING(geno)
closefn.gds(geno)
kin2gds(king, gdsfile)
gds <- openfn.gds(gdsfile)
gds
closefn.gds(gds)
```

---

kingToMatrix

*Convert KING text output to an R Matrix*

---

**Description**

`kingToMatrix` is used to extract the pairwise kinship coefficient estimates from the output text files of `KING -ibdseg`, `KING -kinship`, or `KING -related` and put them into an R object of class `Matrix`. One use of this matrix is that it can be read by the functions [pcair](#) and [pcairPartition](#).

**Usage**

```
## S4 method for signature 'character'
kingToMatrix(king, estimator = c("PropIBD", "Kinship"), sample.include = NULL, thresh = NULL, verbose = FALSE)
## S4 method for signature 'snpgdsIBDClass'
kingToMatrix(king, sample.include = NULL, thresh = 2^(-11/2), verbose = TRUE)
```

**Arguments**

|                |   |
|----------------|---|
| king           | Output from KING, either a <code>snpGdsIBDClass</code> object from <a href="#">snpGdsIBDKING</a> or a character vector of one or more file names output from the command-line version of KING; see 'Details'. |
| estimator      | Which estimates to read in when using command-line KING output; must be either "PropIBD" or "Kinship"; see 'Details'.   |
| sample.include | An optional vector of <code>sample.id</code> indicating all samples that should be included in the output matrix; see 'Details' for usage.  |
| thresh         | Kinship threshold for clustering samples to make the output matrix sparse block-diagonal. When NULL, no clustering is done. See 'Details'.  |
| verbose        | A logical indicating whether or not to print status updates to the console; the default is TRUE.  |

**Details**

king can be a vector of multiple file names if your KING output is stored in multiple files; e.g. KING –kinship run with family IDs returns a .kin and a .kin0 file for pairs within and not within the same family, respectively.

When reading command-line KING output, the estimator argument is required to specify which estimates to read in. When reading KING –ibdseg output, only "PropIBD" will be available; when reading KING –kinship output, only "Kinship" will be available; when reading KING –related output, both "PropIBD" and "Kinship" will be available - use this argument to select which to read. See the KING documentation for details on each estimator.

sample.include has two primary functions: 1) It can be used to subset the KING output. 2) sample.include can include sample.id not in king; this ensures that all samples will be in the output matrix when reading KING –ibdseg output, which likely does not contain all pairs. When left NULL, the function will determine the list of samples from what is observed in king. It is recommended to use sample.include to ensure all of your samples are included in the output matrix.

thresh sets a threshold for clustering samples such that any pair with an estimated kinship value greater than thresh is in the same cluster. All pairwise estimates within a cluster are kept, even if they are below thresh. All pairwise estimates between clusters are set to 0, creating a sparse, block-diagonal matrix. When thresh is NULL, no clustering is done and all samples are returned in one block. This feature is useful when converting KING –ibdseg or KING –robust estimates to be used as a kinship matrix, if you have a lower threshold that you consider 'related'. This feature should not be used when converting KING –robust estimates to be used as `divobj` in [pcair](#) or [pcairPartition](#), as PC-AiR requires the negative estimates to identify ancestrally divergent pairs.

**Value**

An object of class 'Matrix' with pairwise kinship coefficients by KING –ibdseg or KING –robust for each pair of individuals in the sample. The estimates are on both the upper and lower triangle of the matrix, and the diagonal is arbitrarily set to 0.5. sample.id are set as the column and row names of the matrix.

**Author(s)**

Matthew P. Conomos

## References

Conomos M.P., Miller M., & Thornton T. (2015). Robust Inference of Population Structure for Ancestry Prediction and Correction of Stratification in the Presence of Relatedness. *Genetic Epidemiology*, 39(4), 276-293.

Manichaikul, A., Mychaleckyj, J.C., Rich, S.S., Daly, K., Sale, M., & Chen, W.M. (2010). Robust relationship inference in genome-wide association studies. *Bioinformatics*, 26(22), 2867-2873.

## See Also

[pcair](#) and [pcairPartition](#) for functions that use the output matrix.

## Examples

```
# KING --kinship
file.king <- c(system.file("extdata", "MXL_ASW.kin0", package="GENESIS"),
              system.file("extdata", "MXL_ASW.kin", package="GENESIS"))
KINGmat <- kingToMatrix(file.king, estimator="Kinship")

# KING --ibdseg
file.king <- system.file("extdata", "HapMap.seg", package="GENESIS")
KINGmat <- kingToMatrix(file.king, estimator="PropIBD")

# SNPRelate
library(SNPRelate)
gds <- snpgdsOpen(system.file("extdata", "HapMap_ASW_MXL_geno.gds", package="GENESIS"))
king <- snpgdsIBDKING(gds)
KINGmat <- kingToMatrix(king)
snpgdsClose(gds)
```

---

makeSparseMatrix

*Make a sparse matrix from a dense matrix or a table of pairwise values*

---

## Description

makeSparseMatrix is used to create a sparse block-diagonal matrix from a dense matrix or a table of pairwise values, using a user-specified threshold for sparsity. An object of class Matrix will be returned. A sparse matrix may be useful for fitting the association test null model with [fitNullModel](#) when working with very large sample sizes.

## Usage

```
## S4 method for signature 'data.table'
makeSparseMatrix(x, thresh = NULL, sample.include = NULL, diag.value = NULL,
                verbose = TRUE)
## S4 method for signature 'data.frame'
makeSparseMatrix(x, thresh = NULL, sample.include = NULL, diag.value = NULL,
                verbose = TRUE)
## S4 method for signature 'matrix'
makeSparseMatrix(x, thresh = 2^(-11/2), sample.include = NULL, diag.value = NULL,
                verbose = TRUE)
## S4 method for signature 'Matrix'
makeSparseMatrix(x, thresh = 2^(-11/2), sample.include = NULL, diag.value = NULL,
                verbose = TRUE)
```

**Arguments**

|                |  |
|----------------|--|
| x              | An object to coerce to a sparse matrix. May be of class <code>matrix</code> , <code>Matrix</code> , <code>data.frame</code> , or <code>data.table</code> . When a <code>matrix</code> or <code>Matrix</code> , row and column names should be set to <code>sample.ids</code> ; when a <code>data.frame</code> or <code>data.table</code> , should have 3 columns: <code>ID1</code> , <code>ID2</code> , and <code>value</code> . |
| thresh         | Value threshold for clustering samples to make the output matrix sparse block-diagonal. When <code>NULL</code> , no clustering is done. See 'Details'.   |
| sample.include | An optional vector of <code>sample.id</code> indicating all samples that should be included in the output matrix; see 'Details' for usage.   |
| diag.value     | When <code>NULL</code> (by Default), values for the diagonal of the output matrix should be provided in <code>x</code> . Setting <code>diag.value</code> to a numeric value will make all values on the diagonal of the output matrix that value.  |
| verbose        | A logical indicating whether or not to print status updates to the console; the default is <code>TRUE</code> .   |

**Details**

`sample.include` has two primary functions: 1) It can be used to subset samples provided in `x`. 2) `sample.include` can include `sample.id` not in `x`; these additional samples will be included in the output matrix, with a value of 0 for all off-diagonal elements, and the value provided by `diag.value` for the diagonal elements. When left `NULL`, the function will determine the list of samples from what is observed in `x`.

`thresh` sets a threshold for clustering samples such that any pair with a value greater than `thresh` is in the same cluster. All values within a cluster are kept, even if they are below `thresh`. All values between clusters are set to 0, creating a sparse, block-diagonal matrix. When `thresh` is `NULL`, no clustering is done and all samples are returned in one block. This feature is useful when converting a `data.frame` of kinship estimates to a matrix.

**Value**

An object of class 'Matrix'. Samples may be in a different order than in the input `x`, as samples are sorted by ID or rowname within each block (including within the block of unrelateds).

**Author(s)**

Matthew P. Conomos

**See Also**

[kingToMatrix](#) and [pcrelateToMatrix](#) for functions that use this function.

---

pcair

*PC-AiR: Principal Components Analysis in Related Samples*

---

**Description**

`pcair` is used to perform a Principal Components Analysis using genome-wide SNP data for the detection of population structure in a sample. Unlike a standard PCA, PC-AiR accounts for sample relatedness (known or cryptic) to provide accurate ancestry inference that is not confounded by family structure.



**Usage**

```
## S4 method for signature 'gds.class'
pcair(gdsobj, kinobj = NULL, divobj = NULL,
      kin.thresh = 2^(-11/2), div.thresh = -2^(-11/2),
      unrel.set = NULL,
      sample.include = NULL, snp.include = NULL,
      num.cores = 1L, verbose = TRUE, ...)

## S4 method for signature 'SNPGDSFileClass'
pcair(gdsobj, ...)

## S4 method for signature 'GdsGenotypeReader'
pcair(gdsobj, ...)

## S4 method for signature 'GenotypeData'
pcair(gdsobj, ...)

## S4 method for signature 'SeqVarGDSClass'
pcair(gdsobj, ...)
```

**Arguments**

|                             |  |
|-----------------------------|--|
| <code>gdsobj</code>         | An object providing a connection to a GDS file.  |
| <code>kinobj</code>         | A symmetric matrix of pairwise kinship coefficients for every pair of individuals in the sample: upper and lower triangles must both be filled; diagonals should be self-kinship or set to a non-missing constant value. This matrix is used for partitioning the sample into the 'unrelated' and 'related' subsets. See 'Details' for how this interacts with <code>kin.thresh</code> and <code>unrel.set</code> . IDs for each individual must be set as the column names of the matrix. This matrix may also be provided as a GDS object; see 'Details'.                    |
| <code>divobj</code>         | A symmetric matrix of pairwise ancestry divergence measures for every pair of individuals in the sample: upper and lower triangles must both be filled; diagonals should be set to a non-missing constant value. This matrix is used for partitioning the sample into the 'unrelated' and 'related' subsets. See 'Details' for how this interacts with <code>div.thresh</code> . IDs for each individual must be set as the column names of the matrix. This matrix may be identical to <code>kinobj</code> . This matrix may also be provided as a GDS object; see 'Details'. |
| <code>kin.thresh</code>     | Threshold value on <code>kinobj</code> used for declaring each pair of individuals as related or unrelated. The default value is $2^{(-11/2)} \sim 0.022$ , corresponding to 4th degree relatives. See 'Details' for how this interacts with <code>kinobj</code> .   |
| <code>div.thresh</code>     | Threshold value on <code>divobj</code> used for deciding if each pair of individuals is ancestrally divergent. The default value is $-2^{(-11/2)} \sim -0.022$ . See 'Details' for how this interacts with <code>divobj</code> .   |
| <code>unrel.set</code>      | An optional vector of IDs for identifying individuals that are forced into the unrelated subset. See 'Details' for how this interacts with <code>kinobj</code> .   |
| <code>sample.include</code> | An optional vector of IDs for selecting samples to consider for either set.  |
| <code>snp.include</code>    | An optional vector of snp or variant IDs to use in the analysis.   |
| <code>num.cores</code>      | The number of cores to use.  |
| <code>verbose</code>        | Logical indicator of whether updates from the function should be printed to the console; the default is TRUE.  |
| <code>...</code>            | Additional arguments to pass to <code>snpgdsPCA</code> , such as <code>eigen.cnt</code> to control the number of PCs returned.   |

## Details

The basic premise of PC-AiR is to partition the entire sample of individuals into an ancestry representative 'unrelated subset' and a 'related set', perform standard PCA on the 'unrelated subset', and predict PC values for the 'related subset'.

We recommend using software that accounts for population structure to estimate pairwise kinship coefficients to be used in `kinobj`. Any pair of individuals with a pairwise kinship greater than `kin.thresh` will be declared 'related.' Kinship coefficient estimates from the KING-robust software are typically used as measures of ancestry divergence in `divobj`. Any pair of individuals with a pairwise divergence measure less than `div.thresh` will be declared ancestrally 'divergent'. Typically, `kin.thresh` and `div.thresh` are set to be the amount of error around 0 expected in the estimate for a pair of truly unrelated individuals.

There are multiple ways to partition the sample into an ancestry representative 'unrelated subset' and a 'related subset'. In all of the scenarios described below, the set of all samples is limited to those in `sample.include` when it is specified (i.e. not NULL):

If `kinobj` is specified, `divobj` is specified, and `unrel.set = NULL`, then the PC-AiR algorithm is used to find an 'optimal' partition of all samples (see 'References' for a paper describing the PC-AiR algorithm).

If `kinobj` is specified, `divobj` is specified, and `unrel.set` is specified, then all individuals with IDs in `unrel.set` are forced in the 'unrelated subset' and the PC-AiR algorithm is used to partition the rest of the sample; this is especially useful for including reference samples of known ancestry in the 'unrelated subset'.

If `kinobj = NULL`, `divobj = NULL`, and `unrel.set` is specified, then all individuals with IDs in `unrel.set` are put in the 'unrelated subset' and the rest of the individuals are put in the 'related subset'.

If `kinobj = NULL`, `divobj = NULL`, and `unrel.set = NULL`, then the function will perform a "standard" PCA analysis.

NOTE: `kinobj` and `divobj` may be identical, but both must be specified when using the PC-AiR algorithm to partition samples.

## Value

An object of class 'pcair'. A list including:

|                         |  |
|-------------------------|--|
| <code>vectors</code>    | A matrix of principal components; each column is a principal component. Sample IDs are provided as rownames. The number of PCs returned can be adjusted by supplying the <code>eigen.cnt</code> argument, which is passed to <a href="#">snpgdsPCA</a> . |
| <code>values</code>     | A vector of eigenvalues matching the principal components. These values are determined from the standard PCA run on the 'unrelated subset'.  |
| <code>rels</code>       | A vector of IDs for individuals in the 'related subset'.   |
| <code>unrels</code>     | A vector of IDs for individuals in the 'unrelated subset'.   |
| <code>kin.thresh</code> | The threshold value used for declaring each pair of individuals as related or unrelated.   |
| <code>div.thresh</code> | The threshold value used for determining if each pair of individuals is ancestrally divergent.   |
| <code>sample.id</code>  | A vector of IDs for the samples used in the analysis.  |
| <code>nsamp</code>      | The total number of samples in the analysis.   |
| <code>nsnps</code>      | The total number of SNPs used in the analysis.   |
| <code>varprop</code>    | The variance proportion for each principal component.  |

|        |   |
|--------|---|
| call   | The function call passed to pcair.  |
| method | A character string. Either "PC-AiR" or "Standard PCA" identifying which method was used for computing principal components. "Standard PCA" is used if no relatives were identified in the sample. |

**Author(s)**

Matthew P. Conomos

**References**

Conomos M.P., Miller M., & Thornton T. (2015). Robust Inference of Population Structure for Ancestry Prediction and Correction of Stratification in the Presence of Relatedness. *Genetic Epidemiology*, 39(4), 276-293.

Manichaikul, A., Mychaleckyj, J.C., Rich, S.S., Daly, K., Sale, M., & Chen, W.M. (2010). Robust relationship inference in genome-wide association studies. *Bioinformatics*, 26(22), 2867-2873.

**See Also**

[pcairPartition](#) for a description of the function used by `pcair` that can be used to partition the sample into 'unrelated' and 'related' subsets without performing PCA. [plot.pcair](#) for plotting. [kingToMatrix](#) for creating a matrix of pairwise kinship coefficient estimates from KING output text files that can be used for `kinobj` or `divobj`. [GWASTools](#) for a description of the package containing the following functions: [GenotypeData](#) for a description of creating a `GenotypeData` class object for storing sample and SNP genotype data, [MatrixGenotypeReader](#) for a description of reading in genotype data stored as a matrix, and [GdsGenotypeReader](#) for a description of reading in genotype data stored as a GDS file. Also see [snpgdsBED2GDS](#) in the [SNPRelate](#) package for a description of converting binary PLINK files to GDS. The generic functions [summary](#) and [print](#).

**Examples**

```
# file path to GDS file
gdsfile <- system.file("extdata", "HapMap_ASW_MXL_geno.gds", package="GENESIS")
# read in GDS data
HapMap_geno <- gdsfmt::openfn.gds(gdsfile)
# load saved matrix of KING-robust estimates
data("HapMap_ASW_MXL_KINGmat")
# run PC-AiR
mypcair <- pcair(HapMap_geno, kinobj = HapMap_ASW_MXL_KINGmat,
                 divobj = HapMap_ASW_MXL_KINGmat)
gdsfmt::closefn.gds(HapMap_geno)
```

---

|                |   |
|----------------|---|
| pcairPartition | <i>Partition a sample into an ancestry representative 'unrelated subset' and a 'related subset'</i> |
|----------------|---|

---

**Description**

`pcairPartition` is used to partition a sample from a genetic study into an ancestry representative 'unrelated subset' and a 'related subset'. The 'unrelated subset' contains individuals who are all mutually unrelated to each other and representative of the ancestries of all individuals in the sample, and the 'related subset' contains individuals who are related to someone in the 'unrelated subset'.

**Usage**

```
pcairPartition(kinobj, divobj,
               kin.thresh = 2^(-11/2), div.thresh = -2^(-11/2),
               unrel.set = NULL, sample.include = NULL, verbose = TRUE)
```

**Arguments**

|                |  |
|----------------|--|
| kinobj         | A symmetric matrix of pairwise kinship coefficients for every pair of individuals in the sample: upper and lower triangles must both be filled; diagonals should be self-kinship or set to a non-missing constant value. This matrix is used for partitioning the sample into the 'unrelated' and 'related' subsets. See 'Details' for how this interacts with kin.thresh and unrel.set. IDs for each individual must be set as the column names of the matrix. This matrix may also be provided as a GDS object; see 'Details'.                   |
| divobj         | A symmetric matrix of pairwise ancestry divergence measures for every pair of individuals in the sample: upper and lower triangles must both be filled; diagonals should be set to a non-missing constant value. This matrix is used for partitioning the sample into the 'unrelated' and 'related' subsets. See 'Details' for how this interacts with div.thresh. IDs for each individual must be set as the column names of the matrix. This matrix may be identical to kinobj. This matrix may also be provided as a GDS object; see 'Details'. |
| kin.thresh     | Threshold value on kinobj used for declaring each pair of individuals as related or unrelated. The default value is $2^{(-11/2)} \sim 0.022$ , corresponding to 4th degree relatives. See 'Details' for how this interacts with kinobj.  |
| div.thresh     | Threshold value on divobj used for deciding if each pair of individuals is ancestrally divergent. The default value is $-2^{(-11/2)} \sim -0.022$ . See 'Details' for how this interacts with divobj.  |
| unrel.set      | An optional vector of IDs for identifying individuals that are forced into the unrelated subset. See 'Details' for how this interacts with kinobj.   |
| sample.include | An optional vector of IDs for selecting samples to consider for either set.  |
| verbose        | Logical indicator of whether updates from the function should be printed to the console; the default is TRUE.  |

**Details**

We recommend using software that accounts for population structure to estimate pairwise kinship coefficients to be used in kinobj. Any pair of individuals with a pairwise kinship greater than kin.thresh will be declared 'related.' Kinship coefficient estimates from the KING-robust software are typically used as measures of ancestry divergence in divobj. Any pair of individuals with a pairwise divergence measure less than div.thresh will be declared ancestrally 'divergent'. Typically, kin.thresh and div.thresh are set to be the amount of error around 0 expected in the estimate for a pair of truly unrelated individuals. If unrel.set = NULL, the PC-AiR algorithm is used to find an 'optimal' partition (see 'References' for a paper describing the algorithm). If unrel.set and kinobj are both specified, then all individuals with IDs in unrel.set are forced in the 'unrelated subset' and the PC-AiR algorithm is used to partition the rest of the sample; this is especially useful for including reference samples of known ancestry in the 'unrelated subset'.

For large sample sizes, storing both kinobj and divobj in memory may be prohibitive. Both matrices may be stored in GDS files and provided as gds.class objects. [mat2gds](#) saves matrices in GDS format. Alternatively, kinobj (but not divobj) can be represented as a sparse [Matrix](#) object; see [kingToMatrix](#) and [pcrelateToMatrix](#).

Matrix objects from the [Matrix](#) package are also supported.

**Value**

A list including:

rels                    A vector of IDs for individuals in the 'related subset'.  
unrels                  A vector of IDs for individuals in the 'unrelated subset'.

**Note**

pcairPartition is called internally in the function pcair but may also be used on its own to partition the sample into an ancestry representative 'unrelated' subset and a 'related' subset without performing PCA.

**Author(s)**

Matthew P. Conomos

**References**

Conomos M.P., Miller M., & Thornton T. (2015). Robust Inference of Population Structure for Ancestry Prediction and Correction of Stratification in the Presence of Relatedness. *Genetic Epidemiology*, 39(4), 276-293.

Manichaikul, A., Mychaleckyj, J.C., Rich, S.S., Daly, K., Sale, M., & Chen, W.M. (2010). Robust relationship inference in genome-wide association studies. *Bioinformatics*, 26(22), 2867-2873.

**See Also**

[pcair](#) which uses this function for finding principal components in the presence of related individuals. [kingToMatrix](#) for creating a matrix of kinship coefficient estimates or pairwise ancestry divergence measures from KING output text files that can be used as kinobj or divobj. [kin2gds](#) and [mat2gds](#) for saving kinship matrices to GDS.

**Examples**

```
# load saved matrix of KING-robust estimates
data("HapMap_ASW_MXL_KINGmat")
# partition the sample
part <- pcairPartition(kinobj = HapMap_ASW_MXL_KINGmat,
                      divobj = HapMap_ASW_MXL_KINGmat)
```

---

pcrelate

*PC-Relate: Model-Free Estimation of Recent Genetic Relatedness*

---

**Description**

pcrelate is used to estimate kinship coefficients, IBD sharing probabilities, and inbreeding coefficients using genome-wide SNP data. PC-Relate accounts for population structure (ancestry) among sample individuals through the use of ancestry representative principal components (PCs) to provide accurate relatedness estimates due only to recent family (pedigree) structure.

**Usage**

```

## S4 method for signature 'GenotypeIterator'
pcrelate(gdsobj,
  pcs,
  scale = c('overall', 'variant', 'none'),
  ibd.probs = TRUE,
  sample.include = NULL,
  training.set = NULL,
  sample.block.size = 5000,
  maf.thresh = 0.01,
  maf.bound.method = c('filter', 'truncate'),
  small.samp.correct = FALSE,
  verbose = TRUE)
## S4 method for signature 'SeqVarIterator'
pcrelate(gdsobj,
  pcs,
  scale = c('overall', 'variant', 'none'),
  ibd.probs = TRUE,
  sample.include = NULL,
  training.set = NULL,
  sample.block.size = 5000,
  maf.thresh = 0.01,
  maf.bound.method = c('filter', 'truncate'),
  small.samp.correct = FALSE,
  verbose = TRUE)
samplesGdsOrder(gdsobj, sample.include)
calcISAFBeta(gdsobj,
  pcs,
  sample.include,
  training.set = NULL,
  verbose = TRUE)
pcrelateSampBlock(gdsobj,
  betaobj,
  pcs,
  sample.include.block1,
  sample.include.block2,
  scale = c('overall', 'variant', 'none'),
  ibd.probs = TRUE,
  maf.thresh = 0.01,
  maf.bound.method = c('filter', 'truncate'),
  verbose = TRUE)
correctKin(kinBtwn, kinSelf,
  pcs,
  sample.include = NULL)
correctK2(kinBtwn, kinSelf,
  pcs,
  sample.include = NULL,
  small.samp.correct = FALSE)
correctK0(kinBtwn)

```

**Arguments**

|                                    |  |
|------------------------------------|--|
| <code>gdsobj</code>                | An object of class <code>SeqVarIterator</code> from the package <code>SeqVarTools</code> , or an object of class <code>GenotypeIterator</code> from the package <code>GWASTools</code> , containing the genotype data for the variants and samples to be used for the analysis.  |
| <code>pcs</code>                   | A matrix of principal components (PCs) to be used for ancestry adjustment. Each column represents a PC, and each row represents an individual. IDs for each individual must be set as the row names of the matrix.   |
| <code>scale</code>                 | A character string taking the values 'overall', 'variant', or 'none' indicating how genotype values should be standardized. This should be set to 'overall' (the default) in order to do a PC-Relate analysis; see 'Details' for more information.   |
| <code>ibd.probs</code>             | Logical indicator of whether pairwise IBD sharing probabilities ( $k_0$ , $k_1$ , $k_2$ ) should be estimated; the default is TRUE.  |
| <code>sample.include</code>        | A vector of IDs for samples to include in the analysis. If NULL, all samples in <code>gdsobj</code> are included.  |
| <code>training.set</code>          | An optional vector of IDs identifying which samples to use for estimation of the ancestry effect when estimating individual-specific allele frequencies. If NULL, all samples in <code>sample.include</code> are used. See 'Details' for more information.   |
| <code>sample.block.size</code>     | The number of individuals to read-in/analyze at once; the default value is 5000. See 'Details' for more information.   |
| <code>maf.thresh</code>            | Minor allele frequency threshold; if an individual's estimated individual-specific minor allele frequency at a SNP is less than this value, that individual will either have that SNP excluded from the analysis or have their estimated individual-specific minor allele frequency truncated to this value, depending on <code>maf.bound.method</code> . The default value is 0.01.   |
| <code>maf.bound.method</code>      | How individual-specific minor allele frequency estimates less than <code>maf.thresh</code> are handled. When set to 'filter' (default), SNPs for which an individual's estimated individual-specific minor allele frequency are below <code>maf.thresh</code> are excluded from the analysis for that individual. When set to 'truncate', estimated individual-specific minor allele frequencies below <code>maf.thresh</code> have their value set to <code>maf.thresh</code> . |
| <code>small.samp.correct</code>    | Logical indicator of whether to implement a small sample correction.   |
| <code>verbose</code>               | Logical indicator of whether updates from the function should be printed to the console; the default is TRUE.  |
| <code>betaobj</code>               | Output of <code>calcISAFBeta</code> .  |
| <code>sample.include.block1</code> | A vector of IDs for samples to include in block 1.   |
| <code>sample.include.block2</code> | A vector of IDs for samples to include in block 2.   |
| <code>kinBtw</code>                | Output of <code>pcrelateSampBlock</code> .   |
| <code>kinSelf</code>               | Output of <code>pcrelateSampBlock</code> .   |

**Details**

The basic premise of PC-Relate is to estimate kinship coefficients, IBD sharing probabilities, and inbreeding coefficients that reflect recent family (pedigree) relatedness by conditioning out genetic

similarity due to distant population structure (ancestry) with ancestry representative principal components (PCs).

It is important that the PCs used in `pcs` to adjust for ancestry are representative of ancestry and NOT family structure, so we recommend using PCs calculated with PC-AiR (see: [pcair](#)).

In order to perform relatedness estimation, allele frequency estimates are required for centering and scaling genotype values. Individual-specific allele frequencies calculated for each individual at each SNP using the PCs specified in `pcs` are used. There are multiple choices for how genotype values are scaled. When `scale` is 'variant', centered genotype values at each SNP are divided by their expected variance under Hardy-Weinberg equilibrium. When `scale` is 'overall', centered genotype values at all SNPs are divided by the average across all SNPs of their expected variances under Hardy-Weinberg equilibrium; this scaling leads to more stable behavior when using low frequency variants. When `scale` is 'none', genotype values are only centered and not scaled; this won't provide accurate kinship coefficient estimates but may be useful for other purposes. Set `scale` to 'overall' to perform a standard PC-Relate analysis; this is the default. If `scale` is set to 'variant', the estimators are very similar to REAP.

The optional input `training.set` allows the user to specify which samples are used to estimate the ancestry effect when estimating individual-specific allele frequencies. Ideally, `training.set` is a set of mutually unrelated individuals. If prior information regarding pedigree structure is available, this can be used to select `training.set`, or if [pcair](#) was used to obtain the PCs, then the individuals in the PC-AiR 'unrelated subset' can be used. If no prior information is available, all individuals should be used.

The `sample.block.size` can be specified to alleviate memory issues when working with very large data sets. If `sample.block.size` is smaller than the number of individuals included in the analysis, then individuals will be analyzed in separate blocks. This reduces the memory required for the analysis, but genotype data must be read in multiple times for each block (to analyze all pairs), which increases the number of computations required.

`calcISAFbeta` and `pcrelateSampBlock` are provided as separate functions to allow parallelization for large sample sizes. `pcrelate` calls both of these functions internally. When calling these functions separately, use `samplesGdsOrder` to ensure the `sample.include` argument is in the same order as the GDS file. Use `correctKin`, `correctK2`, and `correctK0` after all sample blocks have been completed.

## Value

An object of class 'pcrelate'. A list including:

|                      |  |
|----------------------|--|
| <code>kinBtw</code>  | A data.frame of estimated pairwise kinship coefficients and IBD sharing probabilities (if <code>ibd.probs</code> is TRUE). |
| <code>kinSelf</code> | A data.frame of estimated inbreeding coefficients.   |

## Author(s)

Matthew P. Conomos

## References

Conomos M.P., Reiner A.P., Weir B.S., & Thornton T.A. (2016). Model-free Estimation of Recent Genetic Relatedness. *American Journal of Human Genetics*, 98(1), 127-148.

## See Also

[pcrelateToMatrix](#)



**Examples**

```

library(GWASTools)

# file path to GDS file
gdsfile <- system.file("extdata", "HapMap_ASW_MXL_geno.gds", package="GENESIS")
# read in GDS data
HapMap_geno <- GdsGenotypeReader(filename = gdsfile)
# create a GenotypeData class object
HapMap_genoData <- GenotypeData(HapMap_geno)
# load saved matrix of KING-robust estimates
data("HapMap_ASW_MXL_KINGmat")
# run PC-AiR
mypcair <- pcair(HapMap_genoData, kinobj = HapMap_ASW_MXL_KINGmat,
                 divobj = HapMap_ASW_MXL_KINGmat)

# create a GenotypeBlockIterator object
HapMap_genoData <- GenotypeBlockIterator(HapMap_genoData)
# run PC-Relate
mypcrel <- pcrelate(HapMap_genoData, pcs = mypcair$vectors[,1,drop=FALSE],
                   training.set = mypcair$unrels)
head(mypcrel$kinBwn)
head(mypcrel$kinSelf)

grm <- pcrelateToMatrix(mypcrel)
dim(grm)

close(HapMap_genoData)

```

---

|                  |  |
|------------------|--|
| pcrelateToMatrix | <i>Creates a Genetic Relationship Matrix (GRM) of Pairwise Kinship Coefficient Estimates from PC-Relate Output</i> |
|------------------|--|

---

**Description**

pcrelateToMatrix is used to create a genetic relationship matrix (GRM) of pairwise kinship coefficient estimates from the output of pcrelate.

**Usage**

```

## S4 method for signature 'pcrelate'
pcrelateToMatrix(pcrelobj, sample.include = NULL, thresh = NULL, scaleKin = 2,
                 verbose = TRUE)

```

**Arguments**

|                |   |
|----------------|---|
| pcrelobj       | The object containing the output from pcrelate. This should be a list of class pcrelate containing two data.frames; kinSelf with inbreeding coefficient estimates and kinBwn with pairwise kinship coefficient estimates. |
| sample.include | A vector of IDs for samples to be included in the GRM. The default is NULL, which includes all samples in pcrelobj.   |
| thresh         | Kinship threshold for clustering samples to make the output matrix sparse block-diagonal. This thresholding is done after scaling kinship values by scaleKin. When NULL, no clustering is done. See 'Details'.            |

|          |   |
|----------|---|
| scaleKin | Specifies a numeric constant to scale each estimated kinship coefficient by in the GRM. The default value is 2. |
| verbose  | Logical indicator of whether updates from the function should be printed to the console; the default is TRUE.   |

### Details

This function provides a quick and easy way to construct a genetic relationship matrix (GRM) from the output of pcrelate.

thresh sets a threshold for clustering samples such that any pair with an estimated kinship value greater than thresh is in the same cluster. All pairwise estimates within a cluster are kept, even if they are below thresh. All pairwise estimates between clusters are set to 0, creating a sparse, block-diagonal matrix. When thresh is NULL, no clustering is done and all samples are returned in one block. This feature may be useful for creating a sparse GRM when running association tests with very large sample sizes. Note that thresholding is done after scaling kinship values by scaleKin.

### Value

An object of class 'Matrix' with pairwise kinship coefficients.

### Author(s)

Matthew P. Conomos

### References

Conomos M.P., Reiner A.P., Weir B.S., & Thornton T.A. (2016). Model-free Estimation of Recent Genetic Relatedness. *American Journal of Human Genetics*, 98(1), 127-148.

### See Also

[pcrelate](#) for the function that performs PC-Relate.

---

plot.pcair

*PC-AiR: Plotting PCs*

---

### Description

plot.pcair is used to plot pairs of principal components contained in a class 'pcair' object obtained as output from the pcair function.

### Usage

```
## S3 method for class 'pcair'
plot(x, vx = 1, vy = 2, pch = NULL, col = NULL,
      xlim = NULL, ylim = NULL, main = NULL, sub = NULL,
      xlab = NULL, ylab = NULL, ...)
```

**Arguments**

|      |  |
|------|--|
| x    | An object of class 'pcair' obtained as output from the <code>pcair</code> function.  |
| vx   | An integer indicating which principal component to plot on the x-axis; the default is 1.   |
| vy   | An integer indicating which principal component to plot on the y-axis; the default is 2.   |
| pch  | Either an integer specifying a symbol or a single character to be used in plotting points. If NULL, the default is dots for the 'unrelated subset' and + for the 'related subset'. |
| col  | A specification for the plotting color for points. If NULL, the default is black for the 'unrelated subset' and blue for the 'related subset'.                                     |
| xlim | The range of values shown on the x-axis. If NULL, the default shows all points.  |
| ylim | The range of values shown on the y-axis. If NULL, the default shows all points.  |
| main | An overall title for the plot. If NULL, the default specifies which PC-AiR PCs are plotted.  |
| sub  | A sub title for the plot. If NULL, the default is none.  |
| xlab | A title for the x-axis. If NULL, the default specifies which PC-AiR PC is plotted.   |
| ylab | A title for the y-axis. If NULL, the default specifies which PC-AiR PC is plotted.   |
| ...  | Other parameters to be passed through to plotting functions, (see <a href="#">par</a> ).   |

**Details**

This function provides a quick and easy way to plot principal components obtained with the function `pcair` to visualize the population structure captured by PC-AiR.

**Value**

A figure showing the selected principal components plotted against each other.

**Author(s)**

Matthew P. Conomos

**See Also**

[pcair](#) for obtaining principal components that capture population structure in the presence of relatedness. [par](#) for more in depth descriptions of plotting parameters. The generic function [plot](#).

**Examples**

```
# file path to GDS file
gdsfile <- system.file("extdata", "HapMap_ASW_MXL_geno.gds", package="GENESIS")
# read in GDS data
HapMap_geno <- gdsfmt::openfn.gds(gdsfile)
# load saved matrix of KING-robust estimates
data("HapMap_ASW_MXL_KINGmat")
# run PC-AiR
mypcair <- pcair(HapMap_geno, kinobj = HapMap_ASW_MXL_KINGmat,
                 divobj = HapMap_ASW_MXL_KINGmat)
# plot top 2 PCs
plot(mypcair)
```

```
# plot PCs 3 and 4
plot(mypcair, vx = 3, vy = 4)
gdsfmt::closefn.gds(HapMap_genos)
```

---

print.pcair

*PC-AiR: Principal Components Analysis in Related Samples*


---

## Description

Print methods for pcair

## Usage

```
## S3 method for class 'pcair'
print(x, ...)
## S3 method for class 'pcair'
summary(object, ...)
## S3 method for class 'summary.pcair'
print(x, ...)
```

## Arguments

|        |  |
|--------|--|
| object | An object of class 'pcair', i.e. output from the pcair function. |
| x      | An object of class 'pcair', i.e. output from the pcair function. |
| ...    | Further arguments passed to or from other methods.               |

## Author(s)

Matthew P. Conomos

## See Also

[pcair](#) for obtaining principal components that capture population structure in the presence of relatedness.

## Examples

```
# file path to GDS file
gdsfile <- system.file("extdata", "HapMap_ASW_MXL_genos.gds", package="GENESIS")
# read in GDS data
HapMap_genos <- gdsfmt::openfn.gds(gdsfile)
# load saved matrix of KING-robust estimates
data("HapMap_ASW_MXL_KINGmat")
# run PC-AiR
mypcair <- pcair(HapMap_genos, kinobj = HapMap_ASW_MXL_KINGmat,
                 divobj = HapMap_ASW_MXL_KINGmat)
print(mypcair)
summary(mypcair)
gdsfmt::closefn.gds(HapMap_genos)
```

---

sample\_annotation\_1KG *Annotation for 1000 genomes Phase 3 samples*

---

### Description

Annotation for 1000 genomes Phase 3 samples included in the VCF files in "extdata/1KG".

### Usage

```
data(sample_annotation_1KG)
```

### Format

A data.frame with columns:

- sample.id Sample identifier
- Population Population of sample
- sex Sex of sample

### Source

<ftp://ftp-trace.ncbi.nih.gov/1000genomes/ftp>

### References

A global reference for human genetic variation, The 1000 Genomes Project Consortium, Nature 526, 68-74 (01 October 2015) doi:10.1038/nature15393.

---

varCompCI *Variance Component Confidence Intervals*

---

### Description

varCompCI provides confidence intervals for the variance component estimates found using [fitNullModel](#). The confidence intervals can be found on either the original scale or for the proportion of total variability explained.

### Usage

```
varCompCI(nullMObj, prop = TRUE)
```

### Arguments

nullMObj      A null model object returned by [fitNullModel](#).  
 prop          A logical indicator of whether the point estimates and confidence intervals should be returned as the proportion of total variability explained (TRUE) or on the original scale (FALSE).

**Details**

varCompCI takes the object returned by `fitNullModel` as its input and returns point estimates and confidence intervals for each of the random effects variance component estimates. If a kinship matrix or genetic relationship matrix (GRM) was included as a random effect in the model fit using `fitNullModel`, then this function can be used to provide a heritability estimate when `prop` is TRUE.

**Value**

varCompCI prints a table of point estimates and 95% confidence interval limits for each estimated variance component.

**Author(s)**

Matthew P. Conomos

**See Also**

`fitNullModel` for fitting the mixed model and performing the variance component estimation.

**Examples**

```
library(GWASTools)

# file path to GDS file
gdsfile <- system.file("extdata", "HapMap_ASW_MXL_geno.gds", package="GENESIS")
# read in GDS data
HapMap_geno <- GdsGenotypeReader(filename = gdsfile)
# create a GenotypeData class object
HapMap_genoData <- GenotypeData(HapMap_geno)
# load saved matrix of KING-robust estimates
data("HapMap_ASW_MXL_KINGmat")

# run PC-AiR
mypcair <- pcair(HapMap_genoData, kinobj = HapMap_ASW_MXL_KINGmat,
                 divobj = HapMap_ASW_MXL_KINGmat)

# run PC-Relate
HapMap_genoData <- GenotypeBlockIterator(HapMap_genoData, snpBlock=20000)
mypcrel <- pcrelate(HapMap_genoData, pcs = mypcair$vectors[,1,drop=FALSE],
                  training.set = mypcair$unrels)
close(HapMap_genoData)

# generate a phenotype
set.seed(4)
pheno <- 0.2*mypcair$vectors[,1] + rnorm(mypcair$nsamp, mean = 0, sd = 1)

annot <- data.frame(sample.id = mypcair$sample.id,
                  pc1 = mypcair$vectors[,1], pheno = pheno)

# make covariance matrix
cov.mat <- pcrelateToMatrix(mypcrel, verbose=FALSE)[annot$sample.id, annot$sample.id]

# fit the null mixed model
nullmod <- fitNullModel(annot, outcome = "pheno", covars = "pc1", cov.mat = cov.mat)

# find the variance component CIs
```

```
varCompCI(nullmod, prop = TRUE)  
varCompCI(nullmod, prop = FALSE)
```

# Index

- \*Topic **ancestry**
  - pcair, 24
  - print.pcair, 36
- \*Topic **association**
  - admixmap, 3
  - assocTestAggregate, 6
  - assocTestSingle, 11
  - fitNullModel, 15
- \*Topic **datasets**
  - HapMap\_ASW\_MXL\_KINGmat, 20
  - sample\_annotation\_1KG, 37
- \*Topic **heritability**
  - varCompCI, 37
- \*Topic **mixed model**
  - admixmap, 3
  - assocTestSingle, 11
  - fitNullModel, 15
  - varCompCI, 37
- \*Topic **multivariate**
  - pcair, 24
  - print.pcair, 36
- \*Topic **package**
  - GENESIS-package, 2
- \*Topic **relatedness**
  - pcrelate, 29
- \*Topic **robust**
  - pcair, 24
  - pcrelate, 29
  - print.pcair, 36
- \*Topic **variance component**
  - fitNullModel, 15
  - varCompCI, 37
- admixmap, 3, 19
- admixmapMM (GENESIS-defunct), 19
- AnnotatedDataFrame, 15, 16
- assocTestAggregate, 6, 15, 17–19
- assocTestAggregate, GenotypeIterator-method (assocTestAggregate), 6
- assocTestAggregate, SeqVarIterator-method (assocTestAggregate), 6
- assocTestAggregate-methods (assocTestAggregate), 6
- assocTestMM (GENESIS-defunct), 19
- assocTestSeq (GENESIS-defunct), 19
- assocTestSeqWindow (GENESIS-defunct), 19
- assocTestSingle, 3, 4, 11, 15, 17–19
- assocTestSingle, GenotypeIterator-method (assocTestSingle), 11
- assocTestSingle, SeqVarIterator-method (assocTestSingle), 11
- assocTestSingle-methods (assocTestSingle), 11
- calcISAFBeta (pcrelate), 29
- correctK0 (pcrelate), 29
- correctK2 (pcrelate), 29
- correctKin (pcrelate), 29
- family, 16
- fitNullMM (GENESIS-defunct), 19
- fitNullModel, 3, 4, 6, 12, 14, 15, 19, 23, 37, 38
- fitNullModel, AnnotatedDataFrame-method (fitNullModel), 15
- fitNullModel, data.frame-method (fitNullModel), 15
- fitNullModel, GenotypeData-method (fitNullModel), 15
- fitNullModel, ScanAnnotationDataFrame-method (fitNullModel), 15
- fitNullModel, SeqVarData-method (fitNullModel), 15
- fitNullModel-methods (fitNullModel), 15
- fitNullReg (GENESIS-defunct), 19
- GdsGenotypeReader, 13, 27
- GENESIS (GENESIS-package), 2
- GENESIS-defunct, 19
- GENESIS-package, 2
- GenotypeData, 27
- GenotypeIterator, 4, 12, 13, 31
- GRanges, 7
- GRangesList, 7
- GWASTools, 12, 27, 31
- HapMap\_ASW\_MXL\_KINGmat, 20
- kin2gds, 20, 29



- king2mat (GENESIS-defunct), 19
- kingToMatrix, 3, 19, 21, 21, 24, 27–29
- kingToMatrix, character-method  
(kingToMatrix), 21
- kingToMatrix, snpgdsIBDClass-method  
(kingToMatrix), 21
  
- makeSparseMatrix, 23
- makeSparseMatrix, data.frame-method  
(makeSparseMatrix), 23
- makeSparseMatrix, data.table-method  
(makeSparseMatrix), 23
- makeSparseMatrix, Matrix-method  
(makeSparseMatrix), 23
- makeSparseMatrix, matrix-method  
(makeSparseMatrix), 23
- makeSparseMatrix-methods  
(makeSparseMatrix), 23
- mat2gds, 28, 29
- mat2gds (kin2gds), 20
- Matrix, 15, 28
- MatrixGenotypeReader, 13, 27
- mcols, 7
  
- NcdfGenotypeReader, 13
- nullModelInvNorm (fitNullModel), 15
  
- par, 35
- pcair, 3, 21–23, 24, 29, 32, 35, 36
- pcair, gds.class-method (pcair), 24
- pcair, GdsGenotypeReader-method (pcair),  
24
- pcair, GenotypeData-method (pcair), 24
- pcair, SeqVarGDSCClass-method (pcair), 24
- pcair, SNPGRSFileClass-method (pcair), 24
- pcair-methods (pcair), 24
- pcairPartition, 3, 21–23, 27, 27
- pcrelate, 3, 19, 29, 34
- pcrelate, GenotypeData-method  
(GENESIS-defunct), 19
- pcrelate, GenotypeIterator-method  
(pcrelate), 29
- pcrelate, SeqVarData-method  
(GENESIS-defunct), 19
- pcrelate, SeqVarIterator-method  
(pcrelate), 29
- pcrelateMakeGRM (GENESIS-defunct), 19
- pcrelateReadInbreed (GENESIS-defunct),  
19
- pcrelateReadKinship (GENESIS-defunct),  
19
- pcrelateSampBlock (pcrelate), 29
- pcrelateToMatrix, 3, 19, 24, 28, 32, 33
- pcrelateToMatrix, pcrelate-method  
(pcrelateToMatrix), 33
- plot, 35
- plot.pcair, 3, 27, 34
- print, 27
- print.pcair, 36
- print.summary.pcair (print.pcair), 36
  
- sample\_annotation\_1KG, 37
- samplesGdsOrder (pcrelate), 29
- SeqVarData, 15
- SeqVarIterator, 4, 7, 8, 12, 14, 31
- SeqVarTools, 7, 12, 31
- SeqVarWindowIterator, 8
- snpgdsBED2GDS, 4, 27
- snpgdsIBDKING, 20–22
- snpgdsPCA, 25, 26
- SNPRelate, 27
- summary, 27
- summary.pcair (print.pcair), 36
  
- varCompCI, 3, 18, 37