Package 'bigQF'

May 7, 2026

Type Package

Title Quadratic Forms in Large Matrices

Version 1.6

Author Thomas Lumley

Maintainer Thomas Lumley <t.lumley@auckland.ac.nz>

Description A computationally-efficient leading-eigenvalue approximation to tail probabilities and quantiles of large quadratic forms, in particular for the Sequence Kernel Association Test (SKAT) used in genomics <doi:10.1002/gepi.22136>. Also provides stochastic singular value decomposition for dense or sparse matrices.

URL https://github.com/tslumley/bigQF

Imports svd, CompQuadForm, Matrix, stats, coxme

Suggests knitr, rmarkdown, SKAT

VignetteBuilder knitr

Depends R (>= 2.10), methods

License GPL-2

NeedsCompilation yes

Repository CRAN

Date/Publication 2021-11-23 10:10:11 UTC

Contents

bigQF-package *Quadratic Forms in Large Matrices*

Description

A computationally-efficient leading-eigenvalue approximation to tail probabilities and quantiles of large quadratic forms, in particular for the Sequence Kernel Association Test (SKAT) used in genomics <doi:10.1002/gepi.22136>. Also provides stochastic singular value decomposition for dense or sparse matrices.

Details

The DESCRIPTION file:

Package:	bigQF
Type:	Package
Title:	Quadratic Forms in Large Matrices
Version:	1.6
Author:	Thomas Lumley
Maintainer:	Thomas Lumley <t.lumley@auckland.ac.nz>
Description:	A computationally-efficient leading-eigenvalue approximation to tail probabilities and quantiles of large qu
URL:	https://github.com/tslumley/bigQF
Imports:	svd, CompQuadForm, Matrix, stats, coxme
Suggests:	knitr, rmarkdown, SKAT
VignetteBuilder:	knitr
Depends:	methods
License:	GPL-2

Index of help topics:

```
SKAT.example            Data example from SKAT package
SKAT.matrixfree         Make 'matrix-free' object for SKAT test
bigQF-package           Quadratic Forms in Large Matrices
famSKAT                 Implicit matrix for family-based SKAT test
pQF                     Tail probabilities for quadratic forms
seigen                  Stochastic singular value decomposition
seqMetaExample          Example data, from seqMeta package
sequence                Simulated human DNA variant sequence
sparse.matrixfree       Make 'matrix-free' object from (sparse) Matrix
```

This package computes tail probabilities for large quadratic forms, with the motivation being the SKAT test used in DNA sequence association studies.

The true distribution is a linear combination of 1-df chi-squared distributions, where the coefficients are the non-zero eigenvalues of the matrix `A` defining the quadratic form $z^T Az$. The package uses an approximation to the distribution consisting of the largest `neig` terms in the linear combination plus the Satterthwaite approximation to the rest of the linear combination.

The main function is pQF, which has options for how to compute the leading eigenvalues (Lanczos-type algorithm or stochastic SVD) and how to compute the linear combination (inverting the characteristic function or a saddlepoint approximation). The Lanczos algorithm is from the svd package; the stochastic SVD can be called directly via ssvd or seigen

Given a square matrix, pQF uses it as A. If the input is a non-square matrix M, then A is crossprod(M). The function can also be used matrix-free, given an object containing functions to compute the product and transpose-product by M. This last option is described in the "matrix-free" vignette. The matrix-free algorithm also uses a randomised estimator to estimate the trace of crossprod(A). The function sparse.matrixfree constructs a object for matrix-free use of pQF from a sparse Matrix object. The algorithms are described in the Lumley et al (2018) reference.

Finally, there are functions specifically for the SKAT family of genomic tests. These take a genotype matrix and an adjustment model as arguments and produce an object that contains the test statistic in its Q component and which can be used as an argument to pQF to extract p-values: SKAT.matrixfree and famSKAT. The vignette "Checking pQF vs SKAT" compares SKAT.matrixfree to the SKAT package and illustrates how it can be used

Author(s)

Thomas Lumley

Maintainer: Thomas Lumley <t.lumley@auckland.ac.nz>

References

Tong Chen, Thomas Lumley (2019) Numerical evaluation of methods approximating the distribution of a large quadratic form in normal variables. Computational Statistics & Data Analysis. 139: 75-81,

Lumley et al. (2018) Sequence kernel association tests for large sets of markers: tail probabilities for large quadratic forms. Genet Epidemiol . 2018 Sep;42(6):516-527. doi: 10.1002/gepi.22136

Nathan Halko, Per-Gunnar Martinsson, Joel A. Tropp (2010) Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. https://arxiv.org/abs/0909.4061.

Lee, S., with contributions from Larisa Miropolsky, and Wu, M. (2015). SKAT: SNP-Set (Sequence) Kernel Association Test. R package version 1.1.2.

Lee, S., Wu, M. C., Cai, T., Li, Y., Boehnke, M., and Lin, X. (2011). Rare-variant association testing for sequencing data with the sequence kernel association test. American Journal of Human Genetics, 89:82-93.

---

famSKAT *Implicit matrix for family-based SKAT test*

Description

Like link{SKAT.matrixfree} but for the family-based test of Chen and colleagues

Usage

```
famSKAT(G, model,...)
## S3 method for class 'lmekin'
famSKAT(G, model, kinship,  weights = function(maf) dbeta(maf, 1, 25),...)
## S3 method for class 'GENESIS.nullMixedModel'
famSKAT(G, model, threshold=1e-10,  weights = function(maf) dbeta(maf, 1, 25),...)
## S3 method for class 'famSKAT_lmekin'
update(object,G,...)
## S3 method for class 'famSKAT_genesis'
update(object,G,...)
```

Arguments

G	A 0/1/2 matrix whose columns are markers and whose rows are samples. Should be mostly zero.
model	Object representing a linear mixed model for covariate adjustment. Current methods are for class lmekin which must have been fitted with x=TRUE, and for the output of fitNullMM from the GENESIS package.
kinship	The sparse kinship matrix: the model will have used 2*kinship in its varlist argument.
threshold	A threshold for setting elements of the phenotype precision matrix to exact zeros.
weights	A weight function used in SKAT: the default is the standard one.
object	An existing famSKAT object to be updated with a new set of genotypes (eg for a new gene or genomic window), keeping the same phenotype and kinship structure. Avoids recomputing the Cholesky square root of the phenotype variance matrix, which will often be a computational bottleneck.
...	for future expansion

Value

An object of class c("famSKAT","matrixfree")

Note

The matrix and test statistic both differ by a factor of var(y)/2 from SKAT.matrixfree when used with unrelated individuals (because the Chen et al reference differs from the original SKAT paper by the same factor)

References

Chen H, Meigs JM, Dupuis J (2013) Sequence Kernel Association Test for Quantitative Traits in Family Samples. Genet Epidemiol. 37(2): 196-204.

See Also

SKAT.matrixfree, pQF

Examples

```
data(seqMetaExample)

m<-coxme::lmekin(y~sex+bmi+(1|id),data=pheno2,varlist=2*kins,
x=TRUE,y=TRUE,method="REML")

#first gene
g1snps<-c("1000001", "1000002", "1000003", "1000004", "1000005", "1000006",
"1000007", "1000008", "1000009", "1000010", "1000012", "1000013",
"1000014", "1000015")
Z2gene1<-Z2[,g1snps]

f<-famSKAT(Z2gene1, m, kins)
Q<-f$Q()
all.equal(Q, 56681.209)
## correct p is 0.742756401
pQF(Q,f,neig=4)

## gene10
g10snps<-as.character(1000017:1000036)
Z2gene10<-Z2[,g10snps]
f10<-update(f, Z2gene10)
Q10<-f10$Q()
all.equal(Q10,164656.19)
pQF(Q10,f10,neig=4)
```

---

pQF *Tail probabilities for quadratic forms*

Description

Computes the upper tail probability for quadratic forms in standard Normal variables, using a leading-eigenvalue approximation that is feasible even for large matrices. Can take advantage of sparse matrices.

Usage

```
pQF(x, M, method = c("ssvd", "lanczos", "satterthwaite"), neig = 100,
  tr2.sample.size = 500, q = NULL,
  convolution.method = c("saddlepoint", "integration"),
  remainder.underflow=c("warn","missing","error"))
```

Arguments

x	Vector of quantiles to compute tail probabilities

`M` If `M` is square, it is the matrix in the quadratic form and is assumed to be symmetric. If it is not square, `crossprod(M)` is the matrix in the quadratic form, although this matrix is never formed explicitly. It can also be an object of class `"matrixfree"` to allow matrix-free multiplication for, eg, sparse matrices; see `SKAT.matrixfree` for an example.

`method` Use stochastic SVD ("ssvd") or a thick-restarted Lanczos algorithm ("lanczos") to extract the leading eigenvalues, or use the naive Satterthwaite approximation ("satterthwaite")

`neig` Number of leading eigenvalues to use

`tr2.sample.size`
When `M` is not square, a randomised estimator for the trace of `crossprod(M)^2` is used. This is the sample size. When `M` is of `"matrixfree"` and does not include the trace of crossprod(M) as a component, this sample size will also be used to compute that. See Hutchinson reference.

`q` Power iteration parameter for the stochastic SVD (see the Halko et al reference)

`convolution.method`
For either the "ssvd" or "lanczos" methods, how the convolution of the leading-eigenvalue terms is performed: by Kuonen's saddlepoint approximation or Davies' algorithm inverting the characteristic function

`remainder.underflow`
How should underflow of the remainder term (see Details) be handled? The default is a warning, with `"error"` an error is thrown, and with `"miss"` the return value will be `NaN`.

Details

Increasing `neig` or `q` will improve the accuracy of approximation. In simulated human sequence data, `neig=100` is satisfactory for 5000 samples and markers. Increasing `q` should help when the singular values of `M` decrease slowly. The default sample size for the randomized trace estimator seems to be large enough.

If the remainder term in the approximation has less than 1 degree of freedom it will be dropped, and `remainder.underflow` controls how this is handled. The approximation will then be anticonservative, but usually not seriously so.

In earlier versions, `method="satterthwaite"` rounded the number of degrees of freedom up to the next integer; it now does not round.

In the current version when `M` is of class `"matrix-free"` and `method="ssvd"`, an improved estimator of the remainder term is used. Instead of using Hutchinson's randomised trace estimator and subtracting the known eigenvalues, we apply the randomised trace estimator after projecting orthogonal to the known eigenvectors. After more evaluation, the improvement is likely to be extended to the other methods for rectangular matrices.

By default, Davies's algorithm is run with a tolerance of `1e-9`. This can be changed by setting, eg, `options(bigQF.davies.threshold=1e-12)`

Value

Vector of upper tail probabilities

Author(s)

Thomas Lumley

References

Lumley et al. (2018) "Sequence kernel association tests for large sets of markers: tail probabilities for large quadratic forms" Genet Epidemiol. 2018 Sep;42(6):516-527. doi: 10.1002/gepi.22136

Tong Chen, Thomas Lumley (2019) Numerical evaluation of methods approximating the distribution of a large quadratic form in normal variables. Computational Statistics & Data Analysis. 139: 75-81,

Thomas Lumley (2017) "How to add chi-squareds" https://notstatschat.rbind.io/2017/12/06/how-to-add-chi-squareds/

Thomas Lumley (2016) "Large quadratic forms" https://notstatschat.rbind.io/2016/09/27/large-quadratic-forms/

Nathan Halko, Per-Gunnar Martinsson, Joel A. Tropp (2010) "Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions" https://arxiv.org/abs/0909.4061.

Hutchinson, M. F. (1990). A stochastic estimator of the trace of the influence matrix for Laplacian smoothing splines. Communications in Statistics - Simulation and Computation, 19(2):433-450.

See Also

ssvd,SKAT.matrixfree

Examples

```
data(sequence)
dim(sequence)

skat<-SKAT.matrixfree(sequence)
skat$trace
pQF(c(33782471,7e7,1e8),skat, n=100)


# Don't run these; they take a few minutes
G<-sequence
wuweights<-function(maf) dbeta(maf,1,25)
tmp<-wuweights(colMeans(G)/2)*t(G)
tildeGt<-t(tmp-rowMeans(tmp))/sqrt(2)
sum(tildeGt^2)

pQF(c(33782471,7e7,1e8), tildeGt, n=100)

H<-crossprod(tildeGt)
pQF(c(33782471,7e7,1e8), H, n=100)
```

seigen *Stochastic singular value decomposition*

Description

Extract the leading eigenvalues of a large matrix and their eigenvectors, using random projections.

Usage

```
ssvd(M, n,U=FALSE, V=FALSE,q=3, p=10)
seigen(M, n, only.values = TRUE, q = 3, symmetric = FALSE, spd = FALSE, p = 10)
```

Arguments

M	A square matrix
n	Number of eigenvalues to extract
U, V	If TRUE return the left (respectively, right) singular vectors as well as the singular values
only.values	If TRUE, only extract the eigenvalues, otherwise also extract corresponding eigenvectors
q	Number of power iterations to use in constructing the projection basis (zero or more)
symmetric	If TRUE, assume the matrix is symmetric
spd	If TRUE, assume the matrix is positive definite and use the Nystrom method to improve estimation.
p	The oversampling parameter: number of extra dimensions above n for the random projection

Details

The parameters p and q are as in the reference. Both functions use Algorithm 4.3 to construct a projection; ssvd then uses Algorithm 5.1. With spd=TRUE, seigen uses Algorithm 5.5, otherwise Algorithm 5.3

Value

A list with components

values	eigenvalues
vectors	matrix whose columns are the corresponding eigenvectors

Note

Unlike the Lanczos-type algorithms, this is accurate only for large matrices.

Author(s)

Thomas Lumley

References

Nathan Halko, Per-Gunnar Martinsson, Joel A. Tropp (2010) "Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions" https://arxiv.org/abs/0909.4061.

See Also

ssvd, eigen

Examples

```
data(sequence)
G<-sequence[1:1000,]
H<-tcrossprod(G)
seigen(H,n=10,spd=TRUE,q=5)


eigen(H, symmetric=TRUE,only.values=TRUE)$values[1:10]
```

---

seqMetaExample *Example data, from seqMeta package*

Description

Contains simulated data for two cohorts taken from the seqMeta package. The individual genes are too small for this to be a good use of the leading-eigenvalue approximation, but the data at least allow basic numerical comparisons.

Usage

```
data(seqMetaExample)
```

Format

This contains simulated data for two cohorts

`Z1`,`Z2` Genotype matrices for cohorts 1 and 2 respectively

`pheno1`,`pheno2` phenotype matrices for cohorts 1 and 2 respectively

kins The kinship matrix for cohort 2

Source

https://github.com/DavisBrian/seqMeta

Examples

```
data(seqMetaExample)

m<-coxme::lmekin(y~sex+bmi+(1|id),data=pheno2,varlist=2*kins,
x=TRUE,y=TRUE,method="REML")

#first gene
g1snps<-c("1000001", "1000002", "1000003", "1000004", "1000005", "1000006",
"1000007", "1000008", "1000009", "1000010", "1000012", "1000013",
"1000014", "1000015")
Z2gene1<-Z2[,g1snps]

f<-famSKAT(Z2gene1, m, kins)
Q<-f$Q()
all.equal(Q, 56681.209)
## correct p is 0.742756401
pQF(Q,f,neig=4)
```

---

sequence *Simulated human DNA variant sequence*

Description

A matrix with number of copies of the minor allele for 4028 variants on 5000 people, simulated using the Markov Coalescent Simulator of Chen and coworkers.

Usage

```
data("sequence")
```

Format

The format is: num [1:5000, 1:4028] 0 0 0 0 0 0 0 0 0 0 ...

Source

https://github.com/gchen98/macs

References

Gary K Chen, Paul Marjoram, and Jeffrey D. Wall (2009) Fast and flexible simulation of DNA sequence data Genome Research 19:136-142. http://genome.cshlp.org/content/19/1/136

Examples

```
data(sequence)
summary(colMeans(sequence))
```

SKAT.example *Data example from SKAT package*

Description

These data (probably synthetic) come from the SKAT package. The data set is too small for the leading-eigenvalue approximation to really make sense, but it provides some numerical comparison. The SKAT Q statistic should match exactly, the p-values should be fairly close.

Usage

```
data("SKAT.example")
```

Format

SKAT.example contains the following objects:

Z a numeric genotype matrix of 2000 individuals and 67 SNPs. Each row represents a different individual, and each column represents a different SNP marker.

X a numeric matrix of 2 covariates.

y.c a numeric vector of continuous phenotypes.

y.b a numeric vector of binary phenotypes.

Source

https://www.hsph.harvard.edu/skat/

Examples

```
data(SKAT.example)

skat1mf <- SKAT.matrixfree(SKAT.example$Z)
Q<-skat1mf$Q(SKAT.example$y.c)
all.equal(as.numeric(Q), 234803.786)
## correct value is 0.01874576
pQF(Q, skat1mf, neig=4, convolution.method="integration")

skat2mf <- SKAT.matrixfree(SKAT.example$Z, model=lm(y.c~1, data=SKAT.example))
Q<-skat2mf$Q()
all.equal(Q, 234803.786)
## correct value is 0.01874576
pQF(Q, skat2mf, neig=4, convolution.method="integration")

skat3mf <- SKAT.matrixfree(SKAT.example$Z, model=lm(y.c~X, data=SKAT.example))
Q<-skat3mf$Q()
all.equal(Q, 298041.542)
## correct value is 0.002877041
pQF(Q, skat3mf, neig=4, convolution.method="integration")
```

SKAT.matrixfree *Make 'matrix-free' object for SKAT test*

Description

'Matrix-free' or 'implicit' linear algebra uses a matrix only through the linear operations of multiplying by the matrix or its transpose. It's suitable for sparse matrices, and also for structured matrices that are not sparse but still have fast algorithms for multiplication. The Sequence Kernel Association Test is typically performed on sparse genotype data, but the matrix involved in computations has been centered and is no longer sparse.

Usage

```
SKAT.matrixfree(G,weights=function(maf) dbeta(maf,1,25), model=NULL,...)
## S3 method for class 'lm'
SKAT.matrixfree(G,weights=function(maf) dbeta(maf,1,25), model=NULL,...)
## S3 method for class 'glm'
SKAT.matrixfree(G,weights=function(maf) dbeta(maf,1,25), model=NULL,...)
## S3 method for class 'lmekin'
SKAT.matrixfree(G,weights=function(maf) dbeta(maf,1,25), model=NULL, kinship,...)
```

Arguments

G	A 0/1/2 matrix whose columns are markers and whose rows are samples. Should be mostly zero.
weights	A weight function used in SKAT: the default is the standard one.
model	A linear model, generalised linear model, or lmekin object used for adjustment, or NULL
kinship	A (sparse) kinship matrix. The model will have used 2*kinship in its varlist argument.
...	to keep CMD check happy

Details

If the adjustment model is NULL the object contains the trace of the underlying quadratic form, if the adjustment model is not NULL the trace will be estimated using Hutchinson's randomised estimator inside pQF. The lmekin method calls famSKAT.

Value

An object of class matrixfree

Author(s)

Thomas Lumley

References

Lee, S., with contributions from Larisa Miropolsky, and Wu, M. (2015). SKAT: SNP-Set (Sequence) Kernel Association Test. R package version 1.1.2.

Lee, S., Wu, M. C., Cai, T., Li, Y., Boehnke, M., and Lin, X. (2011). Rare-variant association testing for sequencing data with the sequence kernel association test. American Journal of Human Genetics, 89:82-93.

Wu, M. C., Kraft, P., Epstein, M. P., Taylor, D. M., Channock, S. J., Hunter, D. J., and Lin, X. (2010). Powerful SNP set analysis for case-control genome-wide association studies. American Journal of Human Genetics, 86:929-942.

See Also

pQF sparse.matrixfree

Examples

```
data(sequence)
skat<-SKAT.matrixfree(sequence)
skat$trace
pQF(c(33782471,7e7,1e8), skat,n=100, tr2.sample.size=500)

data(SKAT.example)

skat1mf <- SKAT.matrixfree(SKAT.example$Z)
(Q<-skat1mf$Q(SKAT.example$y.c))
all.equal(as.numeric(Q), 234803.786)

## correct value is 0.01874576
pQF(Q, skat1mf, neig=4, convolution.method="integration")


skat3mf <- SKAT.matrixfree(SKAT.example$Z, model=lm(y.c~X, data=SKAT.example))
(Q<-skat3mf$Q())
all.equal(Q, 298041.542)

## correct value is 0.002877041
pQF(Q, skat3mf, neig=4, convolution.method="integration")
```

---

sparse.matrixfree *Make 'matrix-free' object from (sparse) Matrix*

Description

Packages a matrix (which will typically be a sparse Matrix) for 'matrix-free' or 'implicit' stochastic SVD.

Usage

```
sparse.matrixfree(M)
```

Arguments

M	A matrix or Matrix

Value

Object of class 'matrixfree', with components

mult	Function to multiply by M
tmult	Function to multiply by t(M)
trace	trace of t(M)%*%M, needed for pQF
ncol	dimensions of M
nrow	dimensions of M

Author(s)

Thomas Lumley

See Also

pQF,link{seigen},ssvd, SKAT.matrixfree

Examples

```
data(sequence)
Msp<-sparse.matrixfree(sequence)
ssvd(Msp,n=10)


## this is slow, don't run it
svd(sequence,nu=0,nv=0)$d[1:10]
```

Index