Package 'FastKM'

July 21, 2025

Type Package

Title A Fast Multiple-Kernel Method Based on a Low-Rank Approximation

Version 1.2

Date 2025-05-03

Author Rachel Marceau [aut],
Wenbin Lu [aut],
Michele M. Sale [aut],
Bradford B. Worrall [aut],
Stephen R. Williams [aut],
Fang-Chi Hsu [aut],
Jung-Ying Tzeng [aut],
Shannon T. Holloway [aut, cre]

Maintainer Shannon T. Holloway <shannon.t.holloway@gmail.com>

Description A computationally efficient and statistically rigorous fast Kernel Machine method for multi-kernel analysis. The approach is based on a low-rank approximation to the nuisance effect kernel matrices. The algorithm is applicable to continuous, binary, and survival traits and is implemented using the existing single-kernel analysis software 'SKAT' and 'coxKM'. 'coxKM' can be obtained from <https://github.com/lin-lab/coxKM>.

License GPL-2

Depends rARPACK, stats, methods

Suggests coxKM, SKAT, survival

NeedsCompilation no

Repository CRAN

Date/Publication 2025-05-04 12:10:06 UTC

Contents

`FastKM-package` *A Fast Multiple-Kernel Method Based on a Low-Rank Approximation*

Description

A computationally efficient and statistically rigorous fast Kernel Machine method for multi-kernel analysis. The approach is based on a low-rank approximation to the nuisance effect kernel matrices. The algorithm is applicable to continuous, dichotomous, and survival traits and is implemented using the existing single-kernel analysis software 'SKAT' or 'coxKM'.

Details

Package:	FastKM
Type:	Package
Version:	1.1
Date:	2022-06-05
License:	GPL-2

Author(s)

Shannon T. Holloway, Rachel Marceau, Wenbin Lu, Michele M. Sale, Bradford B. Worrall, Stephen R. Williams, Fang-Chi Hsu, and Jung-Ying Tzeng

Maintainer: Shannon T. Holloway <shannon.t.holloway@gmail.com>

References

Marceau, R., Lu, W., Holloway, S. T., Sale, M. M., Worrall, B. B., Williams, S. R., Hsu, F-C., and Tzeng, J-Y. (2015). A Fast Multiple-Kernel Method with Applications to Detect Gene-Environment Interaction. Genetic Epidemiology, 39, 456-468.

geno *Create a geno Object.*

Description

Creates an object of class geno containing a matrix of one-column-per-marker formated genotype data, the type of kernel, and the weights to be used to create the kernel matrix. This function is used to simplify the input structure of the call to KITdesign.

Usage

```
geno(mat, kernel = "linear", weights = NULL, inheritMode = NA)
```

Arguments

mat	Design matrix for genotype data in either one- or two-column-per-marker format. Object can be a vector, matrix, or data.frame. Missing values can be coded as either 9 or NA.
kernel	Character object indicating type of kernel. Must be one of ('ibs', 'linear', 'quadratic'). For linear kernel, the constant is taken to be zero; for quadratic kernel, it is one.
weights	Weights, if any, to be used in generating kernel. Object can be a vector or matrix. If vector and kernel is not ibs, weights are converted to a diagonal matrix.
inheritMode	For two-column-per-marker format, the inheritance mode to be used to convert to one-column-per-marker format. Must be one of ('add', 'dom', 'rec') indicating additive, dominant, or recessive, respectively.

Details

There are two conventions for genotype data: one- and two-column-per-marker formats. Either format can be used. If genotype data is in one-column format, inheritMode = NA. If genotype data is in two-column format, inheritMode = add/dom/rec.

Non-integer imputed values can be provided. If it is determined that non-integer values are provided and that the kernel was specified as 'ibs,' the kernel will be reset to 'linear.' In addition, if provided in the two-column format, inheritMode will be set to 'add.'

Value

Returns an object of class geno containing the one-column-per-marker genotype data, the weights to be used in generating the kernel, and the type of kernel to be generated.

Author(s)

Shannon T. Holloway

See Also

nongeno, KITdesign

Examples

```
gdata <- matrix(sample(0:1,40,replace=TRUE,prob=c(0.9,0.1)),ncol=4L)

geno(mat = gdata, kernel = "ibs",
     weights = NULL, inheritMode="add")
```

---

geno-class *Class* "geno"

Description

Class contains all information needed to generate a kernel matrix for genotype data.

Objects from the Class

Objects can be created by calls of the form `new("geno", ...)`.

Slots

`mat`: Object of class `"matrix or NULL"`; one-column-per-marker formatted genotype data.

`kernel`: Object of class `"character"`; type of kernel to generate.

`weights`: Object of class `"matrix"`; weights to be used when generating kernel matrix.

Methods

No methods defined with class "geno" in the signature.

Note

This class is used to simplify input and logic within the main program. New objects of this class should be created only through function geno().

Author(s)

Shannon T. Holloway

See Also

geno, nongeno

Examples

```
showClass("geno")
```

KITdesign *A Fast Multiple-Kernel Method Based on a Low-Rank Approximation with Design Matrix Inputs*

Description

A computationally efficient and statistically rigorous fast Kernel Machine method for multi-kernel analysis. The approach is based on a low-rank approximation to the nuisance effect kernel matrices. The algorithm is applicable to continuous, dichotomous, and survival traits and is implemented using the existing single-kernel analysis software 'SKAT' or 'coxKM'. This function accepts as input objects of class geno and/or nongeno containing a design matrix, kernel type, and weights.

Usage

```
KITdesign(y, matA, matB, matC=NULL, x=NULL, trait=NULL, delta=NULL,
          standardize=TRUE, ...)
```

Arguments

y	a vector, matrix, or data.frame of traits. Must be a continuous, dichotomous, or survival trait. Missing values must be coded as NA.
matA	an object of class geno or non-geno that specifies the design matrix, weights, and kernel type for matrix A.
matB	an object of class geno or non-geno that specifies the design matrix, weights, and kernel type for matrix B.
matC	If provided, an object of class non-geno. It is the design matrix of the effect for which the user is testing given the design matrices specified in matA and matB. Missing values are coded as NA. If NULL, Akernel x Bkernel will be used.
x	a vector, matrix, or data.frame. The design matrix of covariates that are not included in either design matrix specified in matA or matB.
trait	One of ('c','d','s', NULL) indicating the type of trait given in input y, where 'c' = continuous, 'd' = dichotomous, and 's' = survival. If NULL, the software will deduce the trait type using the following logic: If input argument delta is not NULL, assume survival. If y is an integer with only two unique values, dichotomous. If y is not an integer or if y is an integer and there are more than two unique values, continuous. Here, integer does not mean the R class but that a numeric is equivalent to its integer truncation.
delta	the status/event indicator in survival analyses. Usually, 0=alive, 1=dead; TRUE/FALSE (TRUE=death); or 1/2 (2 = death). For continuous or dichotomous traits, delta must be NULL.
standardize	If TRUE, input x and all objects of class nongeno will be centered and scaled.
...	Optional arguments to be passed to kernel machine methods and/or coxph.

Details

If missing values are provided in matA, matB, matC, y, or x, individuals with missing values in any of these inputs will be removed from all calculations.

The function SKAT_Null_Model and SKAT of the 'SKAT' R package are used to obtain p-values for continuous and dichotomous traits. For survival traits, Surv and coxph of the 'survival' package and coxKM of the R package 'coxKM' are used. The ellipsis in the call to KITdesign can be used to adjust the default setting of the Kernel Machine methods and coxph. At the time of this documentation, coxph and coxKM have an overlap in argument names, namely 'weights.' If weights are provided by user through the ellipsis, it is assumed that this refers to coxph.

At the time of writing this documentation, the R package 'coxKM' is not available through the CRAN repository, but can be obtained from <https://github.com/lin-lab/coxKM>. This package is only required to be installed if survival traits are analyzed. Only version 0.3 and above of coxKM can be used with this package.

The algorithm maintains the highest possible proportion of variability in both kernel matrices.

Value

A list is returned.

`propA`	The proportion of variability retained in kernel of matA.
`propB`	The proportion of variability retained in kernel of matB.
`pValue`	The p-value as returned by function SKAT or coxKM.

Author(s)

Shannon T. Holloway, Rachel Marceau, Wenbin Lu, Michele M. Sale, Bradford B. Worrall, Stephen R. Williams, Fang-Chi Hsu, and Jung-Ying Tzeng.

References

Marceau, R., Lu, W., Holloway, S. T., Sale, M. M., Worrall, B. B., Williams, S. R., Hsu, F-C., and Tzeng, J-Y. A Fast Multiple-Kernel Method with Applications to Detect Gene-Environment Interaction. Genetic Epidemiology, 39, 456-468.

Examples

```
if( requireNamespace("SKAT", quietly=TRUE) ) {

  matA <- matrix(data = rnorm(100*20), nrow = 100, ncol = 20)
  matB <- matrix(data = rnorm(100), nrow = 100, ncol = 1)

  y <- rnorm(100)

  KITdesign(y = y,
            matA = nongeno(matA, kernel = "linear"),
            matB = nongeno(matB, kernel = "linear"))

}
```

KITkernel *A Fast Multiple-Kernel Method Based on a Low-Rank Approximation with Kernel Inputs.*

Description

A computationally efficient and statistically rigorous fast Kernel Machine method for multi-kernel analysis. The approach is based on a low-rank approximation to the nuisance effect kernel matrices. The algorithm is applicable to continuous, dichotomous, and survival traits and is implemented using the existing single-kernel analysis software 'SKAT' and/or 'coxKM'. This function accepts as input kernel matrices.

Usage

```
KITkernel(y, kmatA, kmatB, kmatC = NULL, x = NULL,
          AkernelC = 0.0, BkernelC = 0.0, trait = NULL, delta = NULL,
          standardize = TRUE, ...)
```

Arguments

y	a vector, matrix, or data.frame of traits. Must be a continuous, dichotomous, or survival trait. Data must be complete.
kmatA	a matrix or data.frame. A kernel matrix. Data must be complete.
kmatB	a matrix or data.frame. A kernel matrix. Data must be complete.
kmatC	If provided, a matrix or data.frame. The kernel matrix of the effect for which the user is testing given matA and matB. Data must be complete. If NULL, the kmatA x kmatB kernel will be used.
x	a vector, matrix, or data.frame. The design matrix of covariates that are not included in either matA or matB, the kernels of which are provided in kmatA and kmatB. Data must be complete.
AkernelC	If kmatC is NULL and kmatA was calculated using a polynomial or interactive kernel, AkernelC is the value of the constant term. For example if kmatA = (1+X^T X), AkernelC = 1.0. This input is used to properly account for the constant terms when generating kmatA x kmatB kernel.
BkernelC	If kmatC is NULL and kmatB was calculated using a polynomial or interactive kernel, BkernelC is the value of the constant term. For example if kmatB = (1+X^T X), BkernelC = 1.0. This input is used to properly account for the constant terms when generating kmatA x kmatB kernel.
trait	One of ('c','d','s', NULL) indicating the type of trait given in input y, where 'c' = continuous, 'd' = dichotomous, and 's' = survival. If NULL, the software will deduce the trait type using the following logic: If input argument delta is not NULL, assume survival. If y is an integer with only two unique values, dichotomous. If y is not an integer or if y is an integer and there are more than two unique values, continuous. Here, integer does not mean the R class but that a numeric is equivalent to its integer truncation.

delta	the status/event indicator in survival analyses. Usually, 0=alive, 1=dead; TRUE/FALSE (TRUE=death); or 1/2 (2 = death). For continuous or dichotomous traits, delta must be NULL.
standardize	If TRUE, input x will be centered and scaled.
...	Optional arguments to be passed to kernel machine methods and/or coxph.

Details

The function SKAT_Null_Model and SKAT of the 'SKAT' R package are used to obtain p-values for continuous and dichotomous traits. For survival traits, Surv and coxph of the 'survival' package and coxKM of the R package 'coxKM' are used. The ellipsis in the call to KITkernel can be used to adjust the default setting of the Kernel Machine methods and coxph. At the time of this documentation, coxph and coxKM have an overlap in argument names, namely 'weights.' If weights are provided by user through the ellipsis, it is assumed that this refers to coxph.

At the time of writing this documentation, the R package 'coxKM' is not available through the CRAN repository, but can be obtained from <https://github.com/lin-lab/coxKM>. This package is only required to be installed if survival traits are analyzed. Only version 0.3 and above of 'coxKM' can be used with this package.

The algorithm maintains the highest possible proportion of variability in both kernel matrices.

Value

A list is returned.

propA	The proportion of variability retained in kmatA.
propB	The proportion of variability retained in kmatB.
pValue	The p-value as returned by function SKAT or coxKM.

Author(s)

Shannon T. Holloway, Rachel Marceau, Wenbin Lu, Michele M. Sale, Bradford B. Worrall, Stephen R. Williams, Fang-Chi Hsu, and Jung-Ying Tzeng.

References

Marceau, R., Lu, W., Holloway, S. T., Sale, M. M., Worrall, B. B., Williams, S. R., Hsu, F-C., and Tzeng, J-Y. A Fast Multiple-Kernel Method with Applications to Detect Gene-Environment Interaction. Genetic Epidemiology, 39, 456-468.

Examples

```
if( requireNamespace("SKAT", quietly=TRUE) ) {

  matA <- matrix(data = rnorm(100*20), nrow = 100, ncol = 20)
  matB <- matrix(data = rnorm(100), nrow = 100, ncol = 1)
  y <- rnorm(100)

  kmatA <- (1 + tcrossprod(matA,matA))^2
  kmatB <- tcrossprod(matB,matB)
```

```
  KITkernel(y = y, kmatA = kmatA, kmatB = kmatB, AkernelC = 1.0)

}
```

---

nongeno *Create a nongeno Object*

Description

Creates an object of class nongeno containing a non-genotype design matrix, the type of kernel to generate, and the weights to be used to create the kernel matrix. This function is used to simplify the input structure of the call to KITdesign.

Usage

```
nongeno(mat, kernel = "linear", weights = NULL)
```

Arguments

`mat`	Design matrix. Missing values must be coded as NA.
`kernel`	Type of kernel. Must be one of ('interactive', 'linear', 'quadratic'). For linear kernel, the constant is taken to be zero; for quadratic kernel, it is one.
`weights`	Weights, if any, to be used in generating kernel. Object can be a vector or matrix. If vector, weights is converted to a diagonal matrix.

Value

Returns an object of class nongeno containing the design matrix, the weights to be used in generating the kernel, and the type of kernel to be generated.

Author(s)

Shannon T. Holloway

See Also

geno, KITdesign

Examples

```
ngdata <- matrix(rnorm(40),ncol=4L)

nongeno(mat = ngdata, kernel = "linear", weights = NULL)
```

nongeno-class *Class* "nongeno"

Description

Class contains all information needed to generate a kernel matrix for non-genotype data.

Objects from the Class

Objects can be created by calls of the form `new("nongeno", ...)`.

Slots

mat: Object of class "matrix or NULL"; design matrix.

kernel: Object of class "character"; type of kernel to generate.

weights: Object of class "matrix"; weights to be used when generating kernel matrix.

Methods

No methods defined with class "nongeno" in the signature.

Note

This class is used to simplify input for and logic within the main program. New objects of this class should be created only through function nongeno().

Author(s)

Shannon T. Holloway

See Also

geno, nongeno

Examples

```
showClass("nongeno")
```

Index