Package 'TransMetaRare'

June 6, 2018

Type Package

Title Trans-ethnic Meta-analysis of Rare Variants in Sequencing Association Studies

Version 0.1

Date 2018-06-05

Author ingchunzi Shi, Seunggeun Lee

Maintainer Jingchunzi (Jing) Shi <shijingc@umich.edu>

Description A score test for rare variant associations in trans-ethnic meta-analysis. The method uses summary level score statistics to carry out gene-based meta-analysis for rare variants.

License GPL (>= 2)

Depends copula, GMMAT, SKAT, mvtnorm, R (>= 3.2.0)

NeedsCompilation no

R topics documented:

TransMeta-Rare-package

Trans-ethnic Meta-analysis of Rare Variants in Sequencing Association Studies

Description

This package implements the score test for rare variant associations in trans-ethnic meta-analysis.

Details

Package:	TransMeta-Rare
Type:	Package
Version:	0.1
Date:	2018-06-05
License:	GPL (>= 2)

This package implements a score test for rare variant associations in trans-ethnic meta-analysis. The method uses summary level score statistics to carry out gene-based meta-analysis for rare variants.

This package depends on the GMMAT package to implement the logistic mixed model for binary traits in the family-based study design. The GMMAT package is currently only available on Linux platform. It can be installed from GitHub:

library(devtools)

install_github("hanchenphd/GMMAT")

Author(s)

Jingchunzi Shi, Seunggeun Lee

Maintainer: Jingchunzi (Jing) Shi <shijingc@umich.edu>

G.list

Example dataset, a list of genotype matrices

Description

a list object of genotypes of all samples. It has 50 elements for 50 genes. Each element is an nxp matrix with n being the total sample size (12,000) over the 4 cohorts and p being the number of SNPs defined in a gene region.

Usage

```
data(G.list)
```

Examples

```
data(G.list)

length(G.list)

dim(G.list[[1]])

colMeans(G.list[[1]])/2
```

`Get_Kernel_Matrix` *A function to estimate the kernel matrix*

Description

Use this function to estimate the kernel matrix which models the correlation structures between the genetic regression coefficients for a one variant across multiple ancestry groups.

Usage

```
Get_Kernel_Matrix(y.list, x.list, G, n.cohort, Group_Idx = NULL,
is.Genetic.Similarity = TRUE)
```

Arguments

`y.list`	a list object for phenotypes. Each element should be a vector of phenotypes. If you have 4 cohorts, it should have 4 elements.
`x.list`	a list object for covariates. Each element should be a vector or a matrix of covariates. If there are 4 cohorts, it should have 4 elements.
`G`	a list object of genotypes of all samples. The number of elements contained in this list should be the total number of gene regions for testing. For example, if testing 50 genes for their associations with the phenotype, then G should have 50 elements for those 50 genes. Each element is an nxp matrix with n being the total sample size over all the cohorts and p being the number of SNPs defined in a gene region.
`n.cohort`	a numeric value of the number of cohort.
`Group_Idx`	a vector of group indicator (default=NULL). If a vector of integers are specified, it assumes causal variants are the same for studies with the same group index, and different for studies with different group indexes. When NULL, studies are assumed to be in different groups with different group indexes.
`is.Genetic.Similarity`	to estimate the genetic similarity kernel or group-wise independence kernel. The default value is TRUE, which tells the function to return the genetic similarity kernel; when the value is FALSE, the function returns the group-wise independence kernel.

Examples

```
### Genetic similarity kernel
data(y.list)
data(x.list)
data(G.list)


Get_Kernel_Matrix(y.list, x.list, G.list, n.cohort = 4)

### Genetic similarity kernel with groups
Get_Kernel_Matrix(y.list, x.list, G.list, n.cohort = 4, Group_Idx=c(1,1,2,3))

### Group-wise independence kernel
Get_Kernel_Matrix(y.list, x.list, G.list, n.cohort = 4, is.Genetic.Similarity = FALSE)
```

K.list *Example dataset, a list of kinship matrices*

Description

a list object of the kinship matrices. It has 4 elements for 4 study cohorts. Each element is a kinship matrix.

Usage

```
data(K.list)
```

Examples

```
data(K.list)

length(K.list)

head(K.list[[1]])
```

TransMeta_Rare_Null_Model *Get parameters and residuals from H0 for population-based study design*

Description

Compute model parameters and residuals under the null model (H0) of no associations for population-based study design. It can be used only when individual level data are available.

Usage

```
TransMeta_Rare_Null_Model(y.list, x.list, n.cohort, out_type = "C",
n.Resampling = 0)
```

Arguments

y.list	a list object for phenotypes. Each element should be a vector of phenotypes. If you have 4 cohorts, it should have 4 elements.
x.list	a list object for covariates. Each element should be a vector or a matrix of covariates. If there are 4 cohorts, it should have 4 elements. If there are no covariates to adjust for, the element should be "intercept". See the examples.
n.cohort	a numeric value of the number of cohort.
out_type	an indicator for the outcome type. "C" for continuous outcomes and "D" for dichotomous outcomes.
n.Resampling	internal use only.

Value

It returns an object that has model parameters and residuals. The returned object will be used to run TransMeta_Rare_wZ.

Examples

```
############################################################
data(y.list)
data(x.list)
data(G.list)

# Compute a p-value of the first gene
Genetic_Kernel = Get_Kernel_Matrix(y.list, x.list, G.list, n.cohort = 4)

obj = TransMeta_Rare_Null_Model(y.list, x.list, n.cohort=4, out_type="C")

TransMeta_Rare_wZ (G.list[[1]], obj, Kernel = Genetic_Kernel)$p.value.normCopula.adj

############################################################
# If you want to use the intercept-only model for the 2nd cohort
x.list[[2]]<-"intercept"

obj = TransMeta_Rare_Null_Model (y.list, x.list, n.cohort=4, out_type="C")

TransMeta_Rare_wZ (G.list[[1]], obj, Kernel = Genetic_Kernel)$p.value.normCopula.adj
```

`TransMeta_Rare_Null_Model_EmmaX` *Get parameters and residuals from H0 for family-based study design*

Description

Compute model parameters and residuals under the null model (H0) of no associations for family-based study design. It can be used only when individual level data are available.

Usage

```
TransMeta_Rare_Null_Model_EmmaX(y.list, x.list, n.cohort, out_type = "C",
K.list = NULL, Kin.File.list = NULL)
```

Arguments

`y.list`	a list object for phenotypes. Each element should be a vector of phenotypes. If you have 4 cohorts, it should have 4 elements.
`x.list`	a list object for covariates. Each element should be a vector or a matrix of covariates. If there are 4 cohorts, it should have 4 elements. If there are no covariates to adjust for, the element should be "intercept". See the examples.
`n.cohort`	a numeric value of the number of cohort.
`out_type`	an indicator for the outcome type. "C" for continuous outcomes and "D" for dichotomous outcomes.
`K.list`	a list object of the kinship matrices. If K.list=NULL, the function reads files in Kin.File.list to obtain kinship matrices.
`Kin.File.list`	a list object of emmax-kin output file names.

Value

It returns an object that has model parameters and residuals. The returned object will be used to run TransMeta_Rare_wZ.

Examples

```
############################################################

data(y.list)
data(x.list)
data(G.list)
data(K.list)

# Compute a p-value of the first gene
Genetic_Kernel = Get_Kernel_Matrix(y.list, x.list, G.list, n.cohort = 4)

obj = TransMeta_Rare_Null_Model_EmmaX(y.list, x.list, n.cohort=4, out_type="C", K.list)
```

```
TransMeta_Rare_wZ (G.list[[1]], obj, Kernel = Genetic_Kernel)$p.value.normCopula.adj
```

`TransMeta_Rare_wZ` *Trans-ethnic meta-analysis with individual level genotype data*

Description

A score test for rare variant associations in trans-ethnic meta-analysis. The method uses summary level score statistics to carry out gene-based meta-analysis for rare variants.

Usage

```
TransMeta_Rare_wZ(Z, obj, combined.weight = FALSE, weights.beta = c(1, 25),
Kernel = Kernel, n.Resampling.Copula = 500, Group_Idx = NULL,
 rho1 = c(0, 1), rho2 = c(0, 0.09, 0.25, 1), impute.method = "fixed",
impute.estimate.maf = 1, missing_cutoff = 0.15)
```

Arguments

`Z`	a numeric genotype matrix with each row as a different individual and each column as a separate snp. Each genotype should be coded as 0, 1, 2, and 9 (or NA) for AA, Aa, aa, and missing, where A is a major allele and a is a minor allele. Missing genotypes will be imputed using observed MAFs.
`obj`	an output object from the TransMeta_Rare_Null_Model or TransMeta_Rare_Null_Model_EmmaX function.
`combined.weight`	a logical value (default=FALSE) for the type of weighting. If it is TRUE, a weight for each SNP is computed using MAFs that are common across studies. If it is FALSE, group specific weights will be used based on group specific MAFs.
`weights.beta`	a numeric vector of parameters of beta weights (default=c(1,25))
`Kernel`	a kernel matrix which models the correlation structures between the genetic regression coefficients for a one variant across multiple ancestry groups. This kernel matrix can be estimated using the "Get_Kernel_Matrix" function.
`n.Resampling.Copula`	number of iterations for the resampling-based copula algorithm, default value is 500.
`Group_Idx`	a vector of group indicator (default=NULL). If a vector of integers are specified, it assumes causal variants are the same for studies with the same group index, and different for studies with different group indexes. When NULL, studies are assumed to be in different groups with different group indexes.

`rho1` value of the ρ_m parameter, default= c(0, 1), can also take value 0 or 1. If rho1 =0, it assumes that the population mean genetic effects among the multiple variants are the same; if rho1 = 1, it assumes that the population mean genetic effects among the multiple variants are independently distributed; rho1 = c(0,1) means we do not make prior assumptions regarding the distribution of the population means.

`rho2` value of the ρ_k parameter, default= c(0, 0.09, 0.25, 1).

`impute.method`
a method to impute missing genotypes (default= "fixed"). "bestguess" imputes missing genotypes as the most likely values(0,1,2), "random" imputes missing genotypes by generating binomial(2,p) random variables (p = MAF), and "fixed" imputes missing genotypes by assigning the mean genotype value (2p).

`impute.estimate.maf`
a numeric value indicating how to estimate MAFs for the imputation. If impute.estimate.maf=1 (default), MetaSKAT uses study-specific MAFs, in which each study MAFs will be used for the imputation. If impute.estimate.maf=2, all samples in the Z matrix will be used to calculate MAFs for the imputation. Previous versions (< ver 0.6) used impute.estimate.maf=2 as a default.

`missing_cutoff`
a cutoff of the missing rates of SNPs (default=0.15). If the first study has SNPs with missing rates higher than the cutoff, these SNPs in the study will be excluded from the analysis. However, the same SNPs in other studies will not be excluded, if their missing rates are lower than the cutoff. The missing rates are calculated study by study.

Details

The rows of Z should be matched with phenotypes and covariates. If there are 4 studies, and study 1,2, 3 and 4 have n1, n2, n3 and n4 samples, the first n1, n2, n3, and n4 rows of Z should be genotypes of the first, second, third and forth studies, respectively. Group_Idx is a vector of group index. Suppose the first two studies are European-based, the third study is Asian-based and the last study is African American-based. If you want to run TransMeta-Rare with assuming ancestry group specific heterogeneity, you can set Group_Idx=c(1,1,2,3), which indicates the first two studies belong to the same group.

Value

`p.value.normCopula.adj`
p-value for the rare variant association in trans-ethnic meta-analysis

`param` estimated parameters of each method.

Examples

```
############################################################

data(y.list)
data(x.list)
data(G.list)
data(K.list)
```

```
#############################################################
# Compute a p-value of the first gene in a population-based study design
Genetic_Kernel = Get_Kernel_Matrix(y.list, x.list, G.list, n.cohort = 4)

obj = TransMeta_Rare_Null_Model(y.list, x.list, n.cohort=4, out_type="C")
TransMeta_Rare_wZ (G.list[[1]], obj, Kernel = Genetic_Kernel)$p.value.normCopula.adj

### groups

Genetic_Kernel_Group = Get_Kernel_Matrix(y.list, x.list, G.list, n.cohort = 4,
Group_Idx=c(1,1,2,3))

TransMeta_Rare_wZ (G.list[[1]], obj, Kernel = Genetic_Kernel_Group,
Group_Idx=c(1,1,2,3))$p.value.normCopula.adj

#############################################################
# Compute a p-value of the first gene in a family-based study design
obj<-TransMeta_Rare_Null_Model_EmmaX(y.list, x.list, n.cohort=4, out_type="C", K.list)

TransMeta_Rare_wZ (G.list[[1]], obj, Kernel = Genetic_Kernel)$p.value.normCopula.adj
```

`x.list` *Example dataset, a list of non-genetic adjusting covariate matrices*

Description

a list object of covariates. It has 4 elements for 4 study cohorts. Each element is a matrix of covariates. The first, third and last elements have two covariates (two columns), and the second element has one covariate (one column).

Usage

```
data(x.list)
```

Examples

```
data(x.list)

length(x.list)

head(x.list[[1]])
```

`y.list` *Example dataset, a list of phenotype vectors*

Description

a list object of continuous phenotypes. It has 4 elements for 4 study cohorts. Each element is a vector of continuous phenotypes

Usage

```
data(y.list)
```

Examples

```
data(y.list)

length(y.list)

length(y.list[[1]])
```

Index