Package 'VariantScan'

June 30, 2022

Version 1.1.9

Date 2022-06-25

Title A Machine Learning Tool for Genetic Association Studies

Description Portable, scalable and highly computationally efficient tool for genetic association studies.``VariantScan'' provides a set of machine learning methods (Linear, Local Polynomial Regression Fitting and Generalized Additive Model with Local Polynomial Smoothing) for genetic association studies that test for disease or trait association with genetic variants (biomarkers, e.g.,genomic (genetic loci), transcriptomic (gene expressions), epigenomic (methylations), proteomic (proteins), metabolomic (metabolites)).
It is particularly useful when local associations and complex nonlinear associations exist.

Maintainer Xinghu Qin <qin.xinghu@163.com>

biocViews

Depends R (>= 3.0)

License GPL (>= 3)

SystemRequirements GNU make

URL https://github.com/xinghuq/VariantScan

BugReports https://github.com/xinghuq/VariantScan/issues

Imports stats,SNPRelate,caret,gam,ModelMetrics

VignetteBuilder knitr

NeedsCompilation no

RoxygenNote 6.1.1

Suggests knitr,testthat,rmarkdown,ggplot2

Author Xinghu Qin [aut, cre, cph] (<https://orcid.org/0000-0003-2351-3610>),
Tianzi Liu [aut],
Peilin Jia [aut]

Repository CRAN

Date/Publication 2022-06-30 11:50:06 UTC

R topics documented:

`gamLoessScan` *Variants (Biomarkers, e.g., genomic (genetic loci), transcriptomic (gene expression), epigenomic (methylations), proteomic(protein), metabolomic (metabolites) variants) Scanning and Association Tests Using Generalized Additive Model with Local Polynomial Regression (LOESS).*

Description

Fitting a Generalized Additive Mixed Models (GAMM) with Local Polynomial Regression in association testing.

Usage

```
gamLoessScan(genotype, traits, U, cv_method = "adaptive_cv",
model_metric = "RMSE", n_hyperparameter_search = 10,verbose=TRUE, ...)
```

Arguments

`genotype`	Varants/genotypes matrix coding in reference allele (0,1,2) or variant count
`traits`	Traits
`U`	Covariates or confounding factors
`cv_method`	Cross-validation
`model_metric`	Model performance metrics, based on which the optimal model is determined.
`n_hyperparameter_search`	Number of hyperparameters for tuning
`verbose`	whether print training messages.
`...`	other arguments passing to generalized additive mixed models (gam)

Details

Fits the specified generalized additive mixed model (GAMM) with LOESS smoothness.

Value

The weights of variants as well as their p-values

References

Wood S.N. (2006b) Generalized Additive Models: An Introduction with R. Chapman and Hall/CRC Press.

Wang, Y. (1998) Mixed effects smoothing spline analysis of variance. J.R. Statist. Soc. B 60, 159-174.

Lin, X and Zhang, D. (1999) Inference in generalized additive mixed models by using smoothing splines. JRSSB. 55(2):381-400.

Examples

```
# not run
f <- system.file('extdata',package='VariantScan')
infile <- file.path(f, "sim1.csv")
geno=read.csv(infile)

traitq=geno[,14]
genotype=geno[,-c(1:14)]
PCs=prcomp(genotype)
test=gamLoessScan(genotype =genotype,traits =(traitq),U=PCs$x[,1:2],n_hyperparameter_search=5)
```

`genmat` *Get genotype matrix from genomic data*

Description

Get genotype matrix from genomic data

Usage

```
genmat(genfile, sample.id = NULL, snp.id = NULL,
snpfirstdim = NA, .snpread = NA, with.id = FALSE, verbose = TRUE, ...)


## S3 method for class 'genmat.bed'
genmat.bed(genfile, sample.id = NULL, snp.id = NULL,
snpfirstdim = NA, .snpread = NA, with.id = FALSE, verbose = TRUE, ...)



## S3 method for class 'genmat.vcf'
genmat.vcf(genfile, sample.id = NULL, snp.id = NULL,
snpfirstdim = NA, .snpread = NA, with.id = FALSE, verbose = TRUE, ...)



## S3 method for class 'genmat.gds'
```

```
genmat.gds(genfile, sample.id = NULL, snp.id = NULL,
snpfirstdim = NA, .snpread = NA, with.id = FALSE, verbose = TRUE, ...)
```

Arguments

genfile	Genetic datasets containg sample ID and SNP ID, format includes bed (plink), vcf, or GDS file.
sample.id	Sample ID
snp.id	SNP ID
snpfirstdim	whether genotypes are stored in the individual-major mode (TRUE), (i.e, list all SNPs for the first individual, and then list all SNPs for the second individual, etc) or (FALSE) for snp-major mode; if NA, determine automatically
.snpread	internal use
with.id	whether return "sample.id" and "snp.id".
verbose	whether printing information
...	more arguments

Details

Effectively get genoptype matrix from various genotype formats, including bed, vcf, or gds.

Value

The function returns an integer matrix with values 0, 1, 2 or NA representing the number of reference allele when with.id=FALSE; or list(genotype, sample.id, snp.id) when with.id=TRUE. The orders of sample and SNP IDs in the genotype matrix are actually consistent with sample.id and snp.id in the GDS file, which may not be as the same as the arguments sampel.id and snp.id specified by users.

References

Zheng, X., & Weir, B. S. (2016). Eigenanalysis of SNP data with an identity by descent interpretation. Theoretical population biology, 107, 65-76.

Examples

```
inp=SNPRelate::snpgdsExampleFileName()

genomat1=genmat.gds(inp)
```

pca *Principal Component Analysis (PCA) on whole genome data with*

Description

Fast implementation of Principal Component Analysis (PCA) on whole genome data

Usage

```
pca(genfile, sample.id = NULL, snp.id = NULL, autosome.only = TRUE,
remove.monosnp = TRUE, maf = NaN, missing.rate = NaN,
algorithm = c("exact", "randomized"),
eigen.cnt = ifelse(identical(algorithm, "randomized"), 16L, 32L),
num.thread = 1L, bayesian = FALSE, need.genmat = FALSE,
genmat.only = FALSE, eigen.method = c("DSPEVX", "DSPEV"),
aux.dim = eigen.cnt * 2L, iter.num = 10L, verbose = TRUE,...)

## S3 method for class 'pca.bed'
pca.bed(genfile, sample.id = NULL, snp.id = NULL, autosome.only = TRUE,
remove.monosnp = TRUE, maf = NaN, missing.rate = NaN,
algorithm = c("exact", "randomized"),
eigen.cnt = ifelse(identical(algorithm, "randomized"), 16L, 32L),
num.thread = 1L, bayesian = FALSE, need.genmat = FALSE,
genmat.only = FALSE, eigen.method = c("DSPEVX", "DSPEV"),
aux.dim = eigen.cnt * 2L, iter.num = 10L, verbose = TRUE,...)

## S3 method for class 'pca.vcf'
pca.vcf(genfile, sample.id = NULL, snp.id = NULL, autosome.only = TRUE,
remove.monosnp = TRUE, maf = NaN, missing.rate = NaN,
algorithm = c("exact", "randomized"),
eigen.cnt = ifelse(identical(algorithm, "randomized"), 16L, 32L),
num.thread = 1L, bayesian = FALSE, need.genmat = FALSE,
genmat.only = FALSE, eigen.method = c("DSPEVX", "DSPEV"),
aux.dim = eigen.cnt * 2L, iter.num = 10L, verbose = TRUE,...)

## S3 method for class 'pca.gds'
pca.gds(genfile, sample.id = NULL, snp.id = NULL, autosome.only = TRUE,
remove.monosnp = TRUE, maf = NaN, missing.rate = NaN,
algorithm = c("exact", "randomized"),
eigen.cnt = ifelse(identical(algorithm, "randomized"), 16L, 32L),
num.thread = 1L, bayesian = FALSE, need.genmat = FALSE,
genmat.only = FALSE, eigen.method = c("DSPEVX", "DSPEV"),
aux.dim = eigen.cnt * 2L, iter.num = 10L, verbose = TRUE,...)
```

Arguments

genfile	Genetic datasets containg sample ID and SNP ID, format includes bed (plink), vcf, or GDS file.
sample.id	a vector of sample id specifying selected samples; if NULL, all samples are used
snp.id	a vector of snp id specifying selected SNPs; if NULL, all SNPs are used
autosome.only	use autosomal SNPs only; if it is a numeric or character value, keep SNPs according to the specified chromosome.
remove.monosnp	remove monomorphic SNPs
maf	filter SNPs with ">= maf" only; if NaN, no MAF threshold
missing.rate	filter the SNPs with "<= missing.rate" only; if NaN, no missing threshold
algorithm	"exact", traditional exact calculation; "randomized", fast PCA with randomized algorithm introduced in Galinsky et al. 2016
eigen.cnt	output the number of eigenvectors; if eigen.cnt <= 0, then return all eigenvectors
num.thread	the number of (CPU) cores used; if NA, detect the number of cores automatically
bayesian	if TRUE, use bayesian normalization
need.genmat	if TRUE, return the genetic covariance matrix
genmat.only	return the genetic covariance matrix only, do not compute the eigenvalues and eigenvectors
eigen.method	"DSPEVX" -compute the top eigen.cnt eigenvalues and eigenvectors using LAPACK::DSPEVX; "DSPEV" -to be compatible with SNPRelate_1.1.6 or earlier, using LAPACK::DSPEV; "DSPEVX" is significantly faster than "DSPEV" if only top principal components are of interest
aux.dim	auxiliary dimension used in fast randomized algorithm
iter.num	iteration number used in fast randomized algorithm
verbose	if TRUE, show information
...	more arguments

Details

Efficient and fast implementation of PCA leveraging the advantage of Genomic Data Structure (GDS) to accelerate computations on SNP data using parallel computing for multi-core symmetric multiprocessing computer architectures. The minor allele frequency and missing rate for each SNP passed in snp.id are calculated over all the samples in sample.id.

Value

Return a of of PCA results, including sample id, SNP id and PCs.

eigenval	eigenvalues
eigenvect	eigenvactors, "# of samples" x "eigen.cnt"
varprop	variance proportion for each principal component

References

Zheng, X., Weir, B. S. (2016). Eigenanalysis of SNP data with an identity by descent interpretation. Theoretical population biology, 107, 65-76.

Patterson N, Price AL, Reich D. (2006). Population structure and eigenanalysis. PLoS Genet.2(12):e190.

Galinsky KJ, Bhatia G, Loh PR, Georgiev S, Mukherjee S, Patterson NJ, Price AL. (2016). Fast Principal-Component Analysis Reveals Convergent Evolution of ADH1B in Europe and East Asia. Am J Hum Genet. 2016 Mar 3;98(3):456-72.

Examples

```
inp=SNPRelate::snpgdsExampleFileName()

pca1=pca.gds(inp, autosome.only=TRUE, remove.monosnp=TRUE, maf=0.01, missing.rate=0.1)
```

VScan *Variants (Biomarkers, e.g., genomic (genetic loci), transcriptomic (gene expression), epigenomic (methylations), proteomic(protein), metabolomic (metabolites) variants) Scanning and Association Tests Using Local Polynomial Fitting (Nonlinear Model) or Linear Model*

Description

Performing association tests for QTLs in genome-wide association studies (GWAS,MWAS,EWAS,PWAS) using nonlinear (Local Polynomial Fitting) or liner model. When a nonlinear model loess smoother is selected, the intercept is used as a null model to test and calculate the R square statistic, then the R square is used as the weight for estimating the variant effect. In linear model, the beta value of the linear model (lm) is used as the weight for estimating the effect size of a variant. This function also applies to case-control studies, where the ROC is used to access the model performance.

Usage

```
VScan(x, y, U=NULL,methods = "loess",span = 0.65, family="gaussian", ...)
```

Arguments

x	Variant matrix, can be genomic (genetic loci), transcriptomic (gene expression), epigenomic (methylations), proteomic(protein), metabolomic (metabolites) variants.
y	Traits
U	Covariates,confounding factors.e.g., age, sex, PCs.
methods	Model fit methods, whether "loess" or "lm". If method is "loess", the model will fit a local polynomial regression, if method is "lm", model will fit a linear model.

span	The local polynomial regression parameter alpha which controls the degree of smoothing
family	The local polynomial regression parameter. if "gaussian", the fitting is done by least-squares, and if "symmetric" a re-descending M estimator is used with Tukey's biweight function. Can be abbreviated.
...	More arguments and parameters passing to loess and loess.control.

Details

Fast association testing and variant scanning (GWAS,MWAS,EWAS,PWAS) using nonlinear (local polynomial fitting, loess) or liner model.When a nonlinear model loess smoother is selected, the intercept is used as a null model to test and calculate the R square statistic, then this R square is used as the weight for estimating the variant effect. In linear model, the beta value of the linear model (lm) is used as the weight for estimating the effect size of a variant.

Biomarkers can be biallelic loci, gene expression, methylations or protein expression. Biallelic markers have only two alleles, in GWAS, genotypes or the reference allele coypes are usually used to test the assocation between phenotypes and genotypes. Most of the cases, linear model is poweful enough to approximate the variant effect. However, if quantitive traits correspond to dynamic gene/protein expression data, and the associations may not be adequately approximated by simple linear model.

Local polynomial regression (LOESS) build on "classical" methods, such as linear and nonlinear least squares regression. They address situations in which the classical procedures do not perform well or cannot be effectively applied without undue labor. LOESS combines much of the simplicity of linear least squares regression with the flexibility of nonlinear regression. It does this by fitting simple models to localized subsets of the data to build up a function that describes the deterministic part of the variation in the data, point by point.

When fitting a local polynomial regression, the model is fitted locally. For the fit at point x, the fit is made using points in a neighbourhood of x, weighted by their distance from x (with differences in "parametric" variables being ignored when computing the distance). The size of the neighbourhood is controlled by alpha (set by span or enp.target). For alpha < 1, the neighbourhood includes proportion alpha of the points, and these have tricubic weighting (proportional to (1 - (dist/maxdist)^3)^3). For alpha > 1, all points are used, with the "maximum distance"" assumed to be alpha^(1/p) times the actual maximum distance for p explanatory variables.

For the default family, fitting is by (weighted) least squares. For family="symmetric" a few iterations of an M-estimation procedure with Tukey's biweight are used. Be aware that as the initial value is the least-squares fit, this need not be a very resistant fit.

Value

Thus function gives the weights and p-values of various QTLs

W	W, the weigts of the variants
p_norm	p values assuming the normal distribution. The W is converted by arcsine transfromaton to normal distribution before estimating the p values
pvalue_chi	p values assuming the Chisquare distribution with df= 1. The W is converted by arcsine transfromaton to normal distribution before estimating the p values

Author(s)

qinxinghu@gmail.com

References

Opsomer, J. D., Ruppert, D. (1997). Fitting a bivariate additive model by local polynomial regression. The Annals of Statistics, 25(1), 186-211.

W. S. Cleveland, E. Grosse and W. M. Shyu (1992) Local regression models. Chapter 8 of Statistical Models in S eds J.M. Chambers and T.J. Hastie, Wadsworth & Brooks/Cole.

Examples

```
# load input data

f <- system.file('extdata',package='VariantScan')
infile <- file.path(f, "sim1.csv")
geno=read.csv(infile)

traitq=geno[,14]
genotype=geno[,-c(1:14)]

# run loess scanning
loessW=VScan(x=genotype,y=(traitq),methods ="loess")

# find association using lm
lmW=VScan(x=genotype,y=(traitq),methods ="lm")

#hightlight the qtl
Loci<-rep("Neutral", 1000)
Loci[c(201,211,221,231,241,251,261,271,281,291)]<-"QT"
Selected_Loci<-Loci[-which(Loci=="Neutral")]

# Plot Manhattan plot
library(ggplot2)
g1=ggplot() +
  geom_point(aes(x=which(Loci=="Neutral"),
  y=-log10(loessW$p_norm[-which(Loci!="Neutral")])), col = "gray83") +
  geom_point(aes(x=which(Loci!="Neutral"),
  y=-log10(loessW$p_norm[-which(Loci=="Neutral")]), colour = Selected_Loci)) +
  xlab("SNPs") + ylab("-log10(p-value)") +ylim(c(0,35))+theme_bw()

g1


g2=ggplot() +
  geom_point(aes(x=which(Loci=="Neutral"),
  y=-log10(lmW$p_norm[-which(Loci!="Neutral")])), col = "gray83") +
  geom_point(aes(x=which(Loci!="Neutral"),
  y=-log10(lmW$p_norm[-which(Loci=="Neutral")]), colour = Selected_Loci)) +
  xlab("SNPs") + ylab("-log10(p-value)") +ylim(c(0,35))+theme_bw()
```

g2

Index