Package 'BATools'

September 30, 2015

Type Package

Title An R Package for Whole Genomic Analysis with Bayesian Models

Version 1.0.0

Date 2015-09-22

Author Chunyu Chen, Wenzhao Yang, Heng Wang ,Robert J. Tempelman

Maintainer Chunyu Chen <chench57@msu.edu>

Depends R (>= 2.15.0), msm, synbreed, coda

Description Bayesian Antedependence Model

License GPL-3

Suggests knitr

VignetteBuilder knitr

R topics documented:

`bafit` *Fitting various Bayesian models*

Description

bafit function can fit various Bayesian models including rrBLUP, BayesA/B/C, antedependence models and etc. Input data can be either a 'baData' object or specify the fixed and random effects seperately.

Usage

```
bafit(dataobj = NULL, op = NULL, y = NULL, Z = NULL, X = NULL,
  trait = NULL)
```

Arguments

dataobj	A list of baData including phenotypes, genotypes and etc.
op	A list of options to run Bayesian models
y	A numeric vector of phenotypes
trait	A string indicating the trait for analysis
A	matrix of genotypes

Value

The result of the analysis as a ba object, a list of estimate of fixed and random effects as well as variance components.

Examples

```
###########Loading and preparing data##########
library(BATools)
data("MSUPRP_sample")
summary(MSUPRP_sample)
pheno<-data.frame(MSUPRP_sample$pheno[,,])
geno<-MSUPRP_sample$geno[,1:500]
ped<-MSUPRP_sample$pedigree
map=MSUPRP_sample$map
sex<-ped$sex
sex<-as.factor(sex)
x<-model.matrix( ~ sex -1,contrasts.arg=list(sex=contrasts(sex, contrasts=F)))
colnames(x)<-c("female","male")
rownames(x)<-ped$ID
pig=create.baData(pheno=pheno,geno=geno,map=map,pedigree=ped,fixed=x,makeAinv=F)
######################BayesA###################
##############Setting up options###############
init=list(df=5,scale=0.01,pi=1)
run_para=list(niter=2000,burnIn=1000,skip=10)
print_mcmc=list(piter=200)
update_para=list(df=FALSE,scale=TRUE,pi=FALSE)
op<-create.options(model="BayesA",method="MCMC",ante=FALSE,priors=NULL,init=init,
               update_para=update_para,run_para=run_para,save.at="BayesA",cv=NULL,print_mcmc=print_mcmc)
#################run BayesA MCMC################
ba<-bafit(dataobj=pig,op=op,trait="driploss")
ba
```

---

baplot *plot result for bafit*

Description

Plot the results for the Bayesian models

Usage

```
baplot(dataobj = NULL, BAout = NULL, y = NULL, Z = NULL, yNa = NULL,
  type = NULL, op = NULL)
```

Arguments

dataobj	baData object
BAout	BAout object contains all the output of the Bayesian model
y	numeric vector of phenotypes
Z	matrix of genotypes
yNa	numeric vector of phenotypes used in cross validation
type	string defines the type of the plot, which can be "pre","man","trace"
	"pre" plot the predicted value against the true value "man" Manhattan plot "trace" Traceplot
op	op object created by create.options that is used for the analysis

Details

You can either provide baData or y and Z for this function. If they're provided at the same time, dataobj will be used for the analysis

Examples

```
baplot(dataobj=pig,BAout=ba,type="pre")
baplot(dataobj=pig,BAout=ba,type="trace",op=op)
```

BATools *BATools*

Description

BATools is an R Package for Whole Genomic Analysis with Bayesian Models

Details

Package:	BATools
Type:	Package
Version:	1.0.0
Date:	2015-09-22
License:	GPL-3

Author(s)

Chunyu Chen, Wenzhao Yang, Heng Wang ,Robert J. Tempelman

Maintainer: Chunyu Chen <chench57@msu.edu>

References

W. Yang, R. J. Tempelman. 2012. A Bayesian Antedependence Model for Whole Genome Prediction. *Genetics*, vol. 190 (4) pp. 1491-1501.

`create.baData` *create* ba *object*

Description

create ba object

Usage

```
create.baData(synbreedobj = NULL, pheno = NULL, geno = NULL, map = NULL,
  pedigree = NULL, family = NULL, covar = NULL, reorderMap = TRUE,
  map.unit = "cM", repeated = NULL, modCovar = NULL, A = NULL,
  Ainv = NULL, fixed = NULL, random = NULL, makeA = FALSE,
  makeAinv = FALSE)
```

Arguments

synbreedobj	A pre-created synbreed data object
pheno	data.frame with individuals organized in rows and traits organized in columns. For unrepeated measures unique rownames should identify individuals. For repeated measures, the first column identifies individuals and a second column indicates repetitions (see also argument repeated).
geno	matrix with individuals organized in rows and markers organized in columns. Genotypes could be coded arbitrarily. Missing values should be coded as NA. Colums or rows with only missing values not allowed. Unique rownames identify individuals and unique colnames markers. If no rownames are available, they are taken from element pheno (if available and if dimension matches). If no colnames are used, the rownames of map are used if dimension matches.
map	data.frame with one row for each marker and two columns (named chr and pos). First columns gives the chromosome (numeric or character but not factor) and second column the position on the chromosome in centimorgan or the physical distance relative to the reference sequence in basepairs. Unique rownames indicate the marker names which should match with marker names in geno. Note that order and number of markers must not be identical with the order in geno. If this is the case, gaps in the map are filled with NA to ensure the same number and order as in element geno of the resulting gpData object.
pedigree	Object of class pedigree.
family	data.frame assigning individuals to families with names of individuals in rownames This information could be used for replacing of missing values with function codeGeno in synbreed.

covar	data.frame with further covariates for all individuals that either appear in pheno, geno or pedigree$ID, e.g. sex or age. rownames must be specified to identify individuals. Typically this element is not specified by the user.
reorderMap	logical. Should markers in geno and map be reordered by chromosome number and position within chromosome according to map (default = TRUE)?
map.unit	Character. Unit of position in map, i.e. 'cM' for genetic distance or 'bp' for physical distance (default = 'cM').
repeated	This column is used to identify the replications of the phenotypic values. The unique values become the names of the third dimension of the pheno object in the gpData. This argument is only required for repeated measurements.
modCovar	vector with colnames which identify columns with covariables in pheno. This argument is only required for repeated measurements.
A	matrix of additive genetic relationship matrix
Ainv	matrix of inverse of A
fixed	matrix or factor of fixed effects
random	matrix or factor of random effects
makeA	logical indicating if create A matrix based on the pedigree information
makeAinv	logical indicating if create A inverse matrix based on the pedigree information, A will be created before creating Ainv is created

Details

The object baData is designed to provide a unified data object for whole genome prediction. It inherits the gpData object from synbreed package and is extended with more optional data item for whole genome prediction. So baData can be used in any functions in synbreed package including data imputation and etc.

Examples

```
#####create baData from gpData######
data("MSUPRP_sample") #load Michigan State University pig data
pheno<-data.frame(MSUPRP_sample$pheno[,,])
geno<-MSUPRP_sample$geno[,1:500]
ped<-MSUPRP_sample$pedigree
map=MSUPRP_sample$map
sex<-ped$sex
sex<-as.factor(sex)
x<-model.matrix( ~ sex -1,contrasts.arg=list(sex=contrasts(sex, contrasts=F)))
colnames(x)<-c("female","male")
rownames(x)<-ped$ID
#create baData object using synbreed object
pig.syn<-create.baData(synbreedobj=MSUPRP_sample,fixed=x,makeAinv=T)
names(pig.syn)
#create baData object using raw data
pig=create.baData(pheno=pheno,geno=geno,map=map,pedigree=ped,fixed=x,makeAinv=F)
```

`create.options` *create* op *object*

Description

Create object for options of bayesian model

Usage

```
create.options(model = NULL, method = NULL, ante = NULL, poly = NULL,
  priors = NULL, init = NULL, D = "V", update_para = NULL,
  run_para = NULL, save.at = NULL, cv = NULL, print_mcmc = NULL,
  ncore = 1, seed = 1, convcrit = 1e-04)
```

Arguments

`model`	string indicate the model for the analysis, `model` can be "GBLUP","BayesA", or "BayesB"
`method`	string indicate the method for the analysis, `model` can be "MCMC" or "EM"
`ante`	logical, must be `TRUE` or `FALSE`
`priors`	`list` contains priors for the Bayesian model, elements in `priors` can be "nu_e","tau2_e", "shape_scale","rate_scale","cdef","alphapi","betapi","mu_m_t","sigma2_m_t", "df_var_t","scale_var_t" `nu_e` `numeric`, prior degrees of freedom for residual variance, default value -1 `tau2_e` `numeric`, prior scale for residual variance,default value 0 `shape_scale` `numeric`, prior shape for scale parameter, default value 0.1 `rate_scale` `numeric`, prior rate for scale parameter, default value 0.1 `cdef` `numeric`, cdef is scale parameter for proposal density on df, default value 0.5 `alphapi` `numeric`, prior α for `pi`, default value 1 `betapi` `numeric`, prior β for `pi`, default value 9 `mu_m_t` `numeric`, prior mean for μ_t, default value 0 `sigma2_m_t` `numeric`, prior variance for μ_t, default value 0.01 `df_var_t` `numeric`, prior degrees of freedom for σ_t^2, default value -1 `scale_var_t` `numeric`, prior scale for σ_t^2, default value 0, the scale and degrees of freedom for σ_t^2 are non-informative priors based on the original paper
`init`	`list` contains initial values for the Bayesian model, elements in `init` can be "df","scale","pi","mut","vart" `df` `numeric`, the starting value of degrees of freedom parameter for SNP effect variance, default value 5 `scale` `numeric`, the starting value of scale parameter for SNP effect variance, default value 0.02 `pi` `numeric`, the starting value of π, which is the percentage of SNP that has variantion to the phenotype, default value 0.1 for BayesB, 1 for other models `mut` `numeric`, the starting value of μ_t, default value 0 `vart` `numeric`, the starting value of σ_t^2, default value 0.5
`D`	`string` indicate use relative variances ("V") or relative precisions ("P"), default is "V"

update_para `list` of `logical` indicate whether a parameter is sampled in the Bayesian model, elements in `update_para` can be "df","scale","pi","mut","vart" `df` `logical`, `TRUE` if degrees of freedom parameter for SNP effect variance needs to be sampled
`scale` `logical`, `TRUE` if scale parameter for SNP effect variance needs to be sampled
`pi` `logical`, `TRUE` if π needs to be sampled
`mut` `logical`, `TRUE` if μ_t needs to be sampled
`vart` `logical`, `TRUE` if σ_t^2 needs to be sampled

run_para `list` elements in `run_para` can be "niter","burnIn","skip" `niter` `numeric`, the number of iterates for MCMC sampling
`burnIn` `numeric`, burnIn period for MCMC sampling
`skip` `numeric` , skip for MCMC sampling

save.at `string` define the directory to save the results

cv k*n `matrix` of `logical`, k is the number of folds and n is the number of observations. `TRUE` indicates the observation is in the training datasets.

print_mcmc `list` define the monitor options for the Bayesian model, elements in `print_mcmc` can be "piter","time_est","print_to"

`piter` numeric, print status every piter. If piter=0, the program don't print any status
`time_est` logical, `TRUE` mean display time left estimates
`print_to` string, "disk" means print to disk in file "log.txt"; "screen" means print to screen

ncore `numeric`, the number of cpu cores for the analysis

seed `numeric`, seed for random number generator

convcrit `numeric`, convergence criteria for EM, default is 1E-4

Examples

```
##################One proper option for rrBLUP REML#####################
init=list(df=NULL,scale=NULL,vare=NULL)
run_para=list(maxiter=100)
update_para=list(df=FALSE,scale=TRUE)
op<-create.options(model="rrBLUP",method="EM",ante=FALSE,priors=NULL,init=init,
   update_para=update_para,run_para=run_para,save.at="rrBLUP",cv=NULL,print_mcmc=NULL,convcrit=1E-4)

##################One proper option for rrBLUP MCMC#####################
init=list(df=5,scale=NULL,vare=NULL)
run_para=list(niter=200000,burnIn=100000,skip=10)
print_mcmc=list(piter=10000)
update_para=list(df=FALSE,scale=TRUE)
op<-create.options(model="rrBLUP",method="MCMC",ante=FALSE,priors=NULL,init=init,
   update_para=update_para,run_para=run_para,save.at="rrBLUP",cv=NULL,print_mcmc=print_mcmc,convcrit=1E-4)

######################One proper option for BayesA MCMC###################
init=list(df=5,scale=0.01,pi=1)
run_para=list(niter=20000,burnIn=10000,skip=10)
print_mcmc=list(piter=2000)
update_para=list(df=FALSE,scale=TRUE,pi=FALSE)
op<-create.options(model="BayesA",method="MCMC",ante=FALSE,priors=NULL,init=init,
   update_para=update_para,run_para=run_para,save.at="BayesA",cv=NULL,print_mcmc=print_mcmc)
```

print.ba *print ba object*

Description

A print method for ba objects

Usage

```
## S3 method for class 'ba'
print(ba)
```

Arguments

ba	an object of the class ba generated by bafit

Value

Null. This function only prints to the screen.

set_init *Set initial values for hyperparameters*

Description

Set initial values for hyperparameters

Usage

```
set_init(x, ...)

## Default S3 method:
set_init(y, Z, df = 5, pi_snp = 0.05, h2 = 0.5,
  c = 1000, model = "rrBLUP", centered = T)

## S3 method for class 'baData'
set_init(baobj, df = 5, pi_snp = 0.05, h2 = 0.5,
  c = 1000, model = "rrBLUP", centered = T, trait = 1)
```

Arguments

y	‘numeric‘ of phenotypes
Z	genotype matrix
df	degrees of freedom parameter
pi_snp	proportion of SNP that have non-zero effect for BayesB and SSVS
h2	hertiability
c	ratio for the smaller variance
model	the model used for analysis, can be ’rrBLUP’, ’BayesA’,’BayesB’, or "SSVS"
centered	logical indicating if Z is centered
baobj	‘baData‘ object

Value

a list of initial values

`std_geno` *Re-center genotype matrix*

Description

Re-center genotype matrix

Usage

```
std_geno(x, ...)

## Default S3 method:
std_geno(x, method = "s")

## S3 method for class 'baData'
std_geno(x, method = "s")
```

Arguments

`x`	genotype matrix
`method`	the method used for centering, 'c' is centering and 's' is standardizing
`baobj`	'baData' object

Value

centered genotype matrix or 'baData' object with centered genotype matrix

Index