Package 'Eagle'

March 3, 2020

Type Package

Title Multiple Locus Association Mapping on a Genome-Wide Scale

Version 2.2

Maintainer Andrew George <andrew.george@csiro.au>

Author Andrew George [aut, cre],
Joshua Bowden [ctb],
Ryan Stephenson [ctb],
Hyun Kang [ctb],
Noah Zaitlen [ctb],
Claire Wade [ctb],
Andrew Kirby [ctb],
David Heckerman [ctb],
Mark Daly [ctb],
Eleazar Eskin [ctb]

Description An implementation of multiple-locus association mapping on a genome-wide scale. 'Eagle' can handle inbred and outbred study populations, populations of arbitrary unknown complexity, and data larger than the memory capacity of the computer. Since 'Eagle' is based on linear mixed models, it is best suited to the analysis of data on continuous traits. However, it can tolerate non-normal data. 'Eagle' reports, as its findings, the best set of snp in strongest association with a trait. For users unfamiliar with R, to perform an analysis, run 'OpenGUI()'. This opens a web browser to the menu-driven user interface for the input of data, and for performing genome-wide analysis.

License GPL-3

Depends R (>= 3.5), shinyFiles, shinyBS, ggplot2, ggthemes

Imports R.utils, mmap, matrixcalc, shiny, shinythemes, shinyjs, stats, utils, parallel, data.table

LinkingTo RcppEigen, Rcpp

LazyData true

ByteCompile TRUE

NeedsCompilation yes

URL http://eagle.r-forge.r-project.org

Contact eaglehelp@csiro.au

Repository CRAN

Date/Publication 2020-03-03 05:20:02 UTC

R topics documented:

`Eagle-package` *Eagle for Genome-wide Association Mapping*

Description

An implementation of multiple-locus association mapping on a genome-wide scale. 'Eagle' can handle inbred and outbred study populations, populations of arbitrary unknown complexity, and data larger than the memory capacity of the computer. Since 'Eagle' is based on linear mixed models, it is best suited to the analysis of data on continuous traits. However, it can tolerate non-normal data. 'Eagle' reports, as its findings, the best set of snp in strongest association with a trait. For users unfamiliar with R, to perform an analysis, run 'OpenGUI()'. This opens a web browser to the menu-driven user interface for the input of data, and for performing genome-wide analysis.

Details

Motivation: Data from genome-wide association studies are analyzed, commonly, with single-locus models. That is, analyzes are performed on a locus-by-locus basis. Multiple-locus approaches that model the association between a trait and multiple loci simultaneously are more powerful. However, these methods do not scale well with study size and many of the packages that implement these methods are not easy to use. Eagle was specifically designed to make genome-wide association mapping with multiple-locus models simple and practical.

Assumptions

1. Individuals are diploid but they can be inbred or outbred.
2. The marker and phenotype data are in separate files.
3. Marker loci are snps. Dominant and multi-allelic loci will need to be converted into biallelic (snp-like) loci.
4. The trait is continuous and normally distributed. Eagle can handle non-normally distributed trait data but there may be a loss of power to detect marker-trait associations.

Important Functions:

1. ReadMarker for reading in the snp data.
2. ReadPheno for reading in the phenotypic data (traits and features/covariates)
3. ReadMap for reading in the marker map.
4. FPR4AM for calculating the value of the gamma parameter to be used by AM that will give a desired false positive rate for detecting SNP-trait associations.
5. AM for performing association mapping on the data.
6. OpenGUI which opens the GUI.

Output: The key output from AM is a list of snp. Each snp identifies a separate genomic region of interest, housing genes that are affecting the trait. Additional summary information such as the size of the snp effects, their statistical significance, and how much phenotypic variation they explain can be obtained by running SummaryAM.

Where to get help: A variety of different help options are available.

- At the R prompt, type

  ```
  library(, "Eagle")
  ```

  for an overview of the package and its functions.
- For detailed help on a function called "foo" say, type

  ```
  help("foo")
  ```
- Visit the Eagle website at http://eagle.r-forge.r-project.org/ where you can find a quick start guide, instructions on getting the most out of Eagle, video tutorials, and other useful information.

Author(s)

Andrew W. George (Data61, CSIRO) with a lot of support from Joshua Bowden (IM&T, CSIRO)

Maintainer: Andrew W. George <andrew.george@csiro.au>

AM *multiple-locus Association Mapping*

Description

AM performs association mapping within a multiple-locus linear mixed model framework. AM finds the best set of marker loci in strongest association with a trait while simultaneously accounting for any fixed effects and the genetic background.

Usage

```
AM(trait = NULL, fformula = NULL, geno = NULL, pheno = NULL,
  map = NULL, Zmat = NULL, ncpu = detectCores(), ngpu = 0,
  quiet = TRUE, maxit = 40, fixit = FALSE, gamma = 1)
```

Arguments

trait	the name of the column in the phenotype data file that contains the trait data. The name is case sensitive and must match exactly the column name in the phenotype data file.
fformula	the right hand side formula for the fixed effects. See below for details. If not specified, only an overall mean will be fitted.
geno	the R object obtained from running ReadMarker. This must be specified.
pheno	the R object obtained from running ReadPheno. This must be specified.
map	the R object obtained from running ReadMap. If not specified, a generic map will be assumed.
Zmat	the R object obtained from running ReadZmat. If not specified, an identity matrix will be assumed.
ncpu	a integer value for the number of CPU that are available for distributed computing. The default is to determine the number of CPU automatically.
ngpu	a integer value for the number of gpu available for computation. The default is to assume there are no gpu available. This option has not yet been implemented.
quiet	a logical value. If set to FALSE, additional runtime output is printed. This is useful for error checking and monitoring the progress of a large analysis.
maxit	an integer value for the maximum number of forward steps to be performed. This will rarely need adjusting.
fixit	a boolean value. If TRUE, then maxit iterations are performed, regardless of the value of the model fit value extBIC. If FALSE, then the model building process is stopped when extBIC increases in value.
gamma	a value between 0 and 1 for the regularization parameter for the extBIC. Values close to 0 lead to an anti-conservative test. Values close to 1 lead to a more conservative test. If this value is left unspecified, a default value of 1 is assumed. See FPR4AM for an empirical approach for setting the gamma value.

Details

This function is used to perform genome-wide association mapping. The phenotypic and SNP data should already be read in prior to running this function (see below for examples). AM builds the linear mixed model iteratively, via forward selection. It is through this model building process that we identify the SNP-trait associations. We use the extended BIC (extBIC) to decide on the 'best' model and when to stop looking for a better model. The conservativeness of extBIC can be adjusted. If the gamma parameter is left at is default setting, then AM is run in its most conservative state (i.e. false positives are minimized but this also decreases the chance of true positives).

When interested in running AM at a certain false positive rate, use FPR4AM. This function uses permutation to find the gamma value for a desired false positive rate for AM.

Below are some examples of how to use AM for genome-wide association mapping of data.

How to perform a basic AM analysis:

Suppose,

- the snp data are contained in the file geno.txt which is a plain space separated text file with no column headings. The file is located in the current working directory. It contains numeric genotype values 0, 1, and 2 for snp genotypes AA, AB, and BB, respectively. It also contains the value X for a missing genotype.
- the phenotype data is contained in the file pheno.txt which is a plain space separated text file containing a single column with the trait data. The first row of the file has the column heading 'y'. This file does not contain any missing data. The file is located in the current working directory.
- there is no map data.

To analyse these data, we would use the following three functions (the parameters can be specified in any order, as well as the functions as long as AM is run last):

```
geno_obj <- ReadMarker(filename='geno.txt', AA=0, AB=1, BB=2, type="text", missing='X')

pheno_obj <- ReadPheno(filename='pheno.txt')

# since gamma is not specified, this will run AM conservatively (where the false positive rate is lowes
res <- AM(trait='y', geno=geno_obj, pheno=pheno_obj)
```

A table of results is printed to the screen and saved in the R object `res`.

How to perform a more complicated AM analysis where the false positive rate is 5%:

Suppose,

- the snp data are contained in the file geno.ped which is a 'PLINK' ped file. See ReadMarker for details. The file is located in /my/dir. Let's assume the file is large, say 50 gigabytes, and our computer only has 32 gigabytes of RAM.
- the phenotype data is contained in the file pheno.txt which is a plain space separated text file with six columns. The first row of the file contains the column headings. The first column is a trait and is labeled y1. The second column is another trait and is labeled y2. The third and fourth columns are nuisance variables and are labeled cov1 and cov2. The fifth and sixth columns are the first two principal components to account for population substructure and are labeled pc1 and pc2. The file contains missing data that are coded as 99. The file is located in /my/dir.
- the map data is contained in the file map.txt, is also located in /my/dir, and the first row has the column headings.
- An 'AM' analysis is performed where the trait of interest is y2, and the fixed effects part of the model is cov1 + cov2 + pc1 + pc2,

To analyse these data, we would run the following:

```
geno_obj <- ReadMarker(filename='/my/dir/geno.ped', type='PLINK', availmemGb=32)

pheno_obj <- ReadPheno(filename='/my/dir/pheno.txt', missing=99)

map_obj   <- ReadMap(filename='/my/dir/map.txt')

# FPR4AM calculates the gamma value corresponding to a desired false positive rate of 5%
ans <- FPR4AM(falseposrate=0.05, numreps=200, trait='y2', fformula=c('cov1 + cov2 + pc1 + pc2'),
           geno=geno_obj, pheno=pheno_obj, map=map_obj)
```

```
# performs association mapping with a 5% false positive rate
res <- AM(trait='y2', fformula=c('cov1 + cov2 + pc1 + pc2'),
          geno=geno_obj, pheno=pheno_obj, map=map_obj,  gamma=ans$setgamma)
```

A table of results is printed to the screen and saved in the R object `res`.

How to perform an analysis where individuals have multiple observations:

Suppose,

- the snp data are contained in the file geno.ped which is a 'PLINK' ped file. See `ReadMarker` for details. The file is located in /my/dir. Let's assume the file is large, say 50 gigabytes, and our computer only has 32 gigabytes of RAM.
- the phenotype data is contained in the file pheno.txt which is a plain space separated text file with six columns. The first row of the file contains the column headings. The first column is a trait and is labeled y1. The second column is another trait and is labeled y2. The third and fourth columns are nuisance variables and are labeled cov1 and cov2. The fifth and sixth columns are the first two principal components to account for population substructure and are labeled pc1 and pc2. The file contains missing data that are coded as 99. The file is located in /my/dir.
- the Z matrix data are contained in the file Zmatrix.txt. The file is located in /my/dir.This file is a design matrix that only contains zeros and ones where each row must contain only a single one in the column that matches the individual's trait value to their corresponding genotype.
- the map data is contained in the file map.txt, is also located in /my/dir, and the first row has the column headings.
- An 'AM' analysis is performed where the trait of interest is y2, and the fixed effects part of the model is cov1 + cov2 + pc1 + pc2.

To analyse these data, we would run the following:

```
geno_obj <-  ReadMarker(filename='/my/dir/geno.ped', type='PLINK', availmemGb=32)

pheno_obj <- ReadPheno(filename='/my/dir/pheno.txt', missing=99)

map_obj   <- ReadMap(filename='/my/dir/map.txt')

Zmat_obj  <- ReadZmat(filename='/my/dir/Zmatrix.txt')

res <- AM(trait='y2', fformula=c('cov1 + cov2 + pc1 + pc2'),
          geno=geno_obj, pheno=pheno_obj, map=map_obj, Zmat=Zmat_obj )
```

A table of results is printed to the screen and saved in the R object `res`.

Dealing with missing marker data:

`AM` can tolerate some missing marker data. However, ideally, a specialized genotype imputation program such as 'BEAGLE' or 'PHASE2', should be used to impute the missing marker data before being read into 'Eagle'.

Dealing with missing trait data:

`AM` deals automatically with individuals with missing trait data. These individuals are removed from the analysis and a warning message is generated.

Dealing with missing explanatory variable values:

AM deals automatically with individuals with missing explanatory variable values. These individuals are removed from the analysis and a warning message is generated

Error Checking:

Most errors occur when reading in the data. However, as an extra precaution, if quiet=FALSE, then additional output is printed during the running of AM. If AM is failing, then this output can be useful for diagnosing the problem.

Value

A list with the following components:

trait: column name of the trait being used by 'AM'.

fformula: the fixed effects part of the linear mixed model.

indxNA_pheno: a vector containing the row indexes of the phenotyic data that have been removed from the analysis.

indxNA_geno: a vector containing the row indexes of those genotypes that have been removed from the analysis due to missing data.

Mrk: a vector with the names of the snp in strongest and significant association with the trait.If no loci are found to be significant, then this component is NA.

Chr: the chromosomes on which the identified snp lie.

Pos: the map positions for the identified snp.

Indx: the column indexes in the marker file of the identified snp.

ncpu: number of cpu used for the calculations.

availmemGb: amount of RAM in gigabytes that has been set by the user.

quiet: boolean value of the parameter.

extBIC: numeric vector with the extended BIC values for the loci found to be in significant association with the trait.

gamma the numeric value of the parameter.

See Also

FPR4AM , ReadMarker, ReadPheno, ReadZmat, and ReadMap

Examples

```
 ## Not run:
# Since the following code takes longer than 5 seconds to run, it has been tagged as dontrun.
 # However, the code can be run by the user.
 #

 #-------------------------
 #  Example
 #------------------------
```

```
  # read the map
  #~~~~~~~~~~~~~~~

  # File is a plain space separated text file with the first row
  # the column headings
  complete.name <- system.file('extdata', 'map.txt',
                                 package='Eagle')
  map_obj <- ReadMap(filename=complete.name)

  # read marker data
  #~~~~~~~~~~~~~~~~~~~~
  # Reading in a PLINK ped file
 # and setting the available memory on the machine for the reading of the data to 8  gigabytes
  complete.name <- system.file('extdata', 'geno.ped',
                                   package='Eagle')
  geno_obj <- ReadMarker(filename=complete.name,  type='PLINK', availmemGb=8)

  # read phenotype data
  #~~~~~~~~~~~~~~~~~~~~~~~

  # Read in a plain text file with data on a single trait and two covariates
  # The first row of the text file contains the column names y, cov1, and cov2.
  complete.name <- system.file('extdata', 'pheno.txt', package='Eagle')

  pheno_obj <- ReadPheno(filename=complete.name)


 # Performing multiple-locus genome-wide association mapping with a model
 #    with fixed effects cov1 and cov2 and an intercept. The intercept
 #    need not be specified as it is assumed.
 #~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

  res <- AM(trait = 'y',
                         fformula=c('cov1+cov2'),
                         map = map_obj,
                         pheno = pheno_obj,
                         geno = geno_obj )

## End(Not run)
```

`FPR4AM` *Set the false positive rate for* `AM`

Description

The gamma parameter in `AM` controls the false positive rate of the model building process. This function uses permutation to find the gamma value for a desired false positive rate.

Usage

```
FPR4AM(falseposrate = 0.05, trait = trait, numreps = 200,
  fformula = NULL, numgammas = 20, geno = NULL, pheno = NULL,
  map = NULL, Zmat = NULL, ncpu = detectCores(), ngpu = 0, seed = 101)
```

Arguments

falseposrate	the desired false positive rate.
trait	the name of the column in the phenotype data file that contains the trait data. The name is case sensitive and must match exactly the column name in the phenotype data file. This parameter must be specified.
numreps	the number of replicates upon which to base the calculation of the false positive rate. We have found 200 replicates to be sufficient but more is better.
fformula	the right hand side formula for the fixed effects part of the model.
numgammas	the number of equidistant gamma values from 0 to 1 for which to calculate the false positive rate of the model building process. This should not need adjusting.
geno	the R object obtained from running ReadMarker. This must be specified.
pheno	the R object obtained from running ReadPheno. This must be specified.
map	the R object obtained from running ReadMap. If not specified, a generic map will be assumed.
Zmat	the R object obtained from running ReadZmat. If not specified, an identity matrix will be assumed.
ncpu	a integer value for the number of CPU that are available for distributed computing. The default is to determine the number of CPU automatically.
ngpu	a integer value for the number of gpu available for computation. The default is to assume there are no gpu available. This option has not yet been implemented.
seed	a integer value for the starting seed for the permutations.

Details

The false positive rate for AM is controlled by its gamma parameter. Values close to 1 (0) decreases (increases) the false positive rate of detecting SNP-trait associations. There is no analytical way of setting gamma for a specified false positive rate. So we are using permutation to do this empirically.

By setting falseposrate to the desired false positive rate, this function will find the corresponding gamma value for AM.

A table of other gamma values for a range of false positive rates is also given.

To increase the precision of the gamma estimates, increase numreps.

Value

A list with the following components:

numreps: the number of permutations performed.

gamma: the vector of gamma values.

falsepos: the false positive rates for the gamma values.

setgamma: the gamma value that gives a false positive rate of falseposrate

See Also

AM

Examples

```
## Not run:
# Since the following code takes longer than 5 seconds to run, it has been tagged as dontrun.
# However, the code can be run by the user.
#

#-------------------------
#  Example
#------------------------

# read the map
#~~~~~~~~~~~~~~

# File is a plain space separated text file with the first row
# the column headings
complete.name <- system.file('extdata', 'map.txt',
                                 package='Lion')
map_obj <- ReadMap(filename=complete.name)

# read marker data
#~~~~~~~~~~~~~~~~~~~~
# Reading in a PLINK ped file
# and setting the available memory on the machine for the reading of the data to 8  gigabytes
complete.name <- system.file('extdata', 'geno.ped',
                                   package='Lion')
geno_obj <- ReadMarker(filename=complete.name,  type='PLINK', availmemGb=8)

# read phenotype data
#~~~~~~~~~~~~~~~~~~~~~~~

# Read in a plain text file with data on a single trait and two covariates
# The first row of the text file contains the column names y, cov1, and cov2.
complete.name <- system.file('extdata', 'pheno.txt', package='Lion')

pheno_obj <- ReadPheno(filename=complete.name)


#  Suppose we want to perform the AM analysis at a 5% false positive rate.
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

ans <- FPR4AM(falseposrate = 0.05,
              trait = 'y',
              fformula=c('cov1+cov2'),
              map = map_obj,
              pheno = pheno_obj,
              geno = geno_obj)
```

```
res <- AM(trait =  'y',
              fformula=c('cov1+cov2'),
              map = map_obj,
              pheno = pheno_obj,
              geno = geno_obj,
              gamma = ans$setgamma)


## End(Not run)
```

OpenGUI *Browser-based Graphical User Interface*

Description

Opens a web browser to act as a user-friendly interface to 'Eagle'

Usage

```
OpenGUI()
```

Details

Once OPenGUI is run, your default web-browser will open automatically the Eagle GUI. We have written our GUI in html code. The GUI allows users to input data files, analyse data, summarise findings, and view results via interactive plots. We have designed the GUI for users whom may be unfamilar with R and wishing to avoid having to write R code.

Note, that even though a web browser is being used as the user interface, everything remains local to the computer.

Examples

```
## Not run:
# opens a web browser
OpenGUI()

## End(Not run)
```

ReadMap *Read map file*

Description

Read in the marker map data.

Usage

```
ReadMap(filename = NULL, csv = FALSE, header = TRUE)
```

Arguments

filename	contains the name of the map file. The file name needs to be in quotes. If the file is not in the working directory, then the full path to the file is required.
csv	a logical value. When TRUE, a csv file format is assumed. When FALSE, a space separated format is assumed.
header	a logical value. When TRUE, the first row of the file contains the column headings.

Details

Association mapping, unlike classical linkage mapping, does not require a map to find marker-trait associations. So, reading in a map file is optional. If a map file is supplied, then the marker names from this file are used when reporting the findings from AM. If a map file is not supplied, then generic names M1, M2, ..., are assigned to the marker loci where the number refers to the column number in the marker file.

A space separated text file with column headings is assumed as the default input. The map file can have three or four columns. If the map file has three columns, then it is assumed that the three columns are the marker locus names, the chromosome number, and the map position (in any units). If the map file has four columns as with a 'PLINK map file, then the columns are assumed to be the marker locus names, the chromosome number, the map position in centimorgans, and the map position in base pairs.

Missing values are allowed but not in the first column of the file (i.e. the marker labels are not allowed to be missing).

The order of the marker loci in this file is assumed to be the same order as the loci in the marker data file.

The first column of the map file is assumed to contain the marker names.

Value

a data frame is returned of the map data.

See Also

ReadMarker and ReadPheno.

Examples

```
# Read in  example map data from ./extdata/

# find the full location of the map data
complete.name <- system.file('extdata', 'map.txt', package='Eagle')

# read in map data
map_obj <- ReadMap(filename=complete.name)

# look at first few rows of the map file
head(map_obj)
```

ReadMarker *Read marker data.*

Description

A function for reading in different types of snp marker data.

Usage

```
ReadMarker(filename = NULL, type = "text", missing = NULL, AA = NULL,
  AB = NULL, BB = NULL, availmemGb = 16, quiet = TRUE)
```

Arguments

filename	contains the name of the marker file. The file name needs to be in quotes. If the file is not in the working directory, then the full path to the file is required.
type	specify the type of file. Choices are 'text' (the default) , PLINK, and vcf.
missing	the number or character for a missing genotype in the text file. There is no need to specify this for a vcf or PLINK ped file. Missing allele values in a vcf file are coded as "." and missing allele values in a PLINK file must be coded as '0' or '-'.
AA	the character or number corresponding to the 'AA' snp genotype in the marker genotype file. This need only be specified if the file type is 'text'. If a character then it must be in quotes.
AB	the character or number corresponding to the 'AB' snp genotype in the marker genotype file. This need only be specified if the file type is 'text'. This can be left unspecified if there are no heterozygous genotypes (i.e. the individuals are inbred). Only a single heterozygous genotype is allowed ('Eagle' does not distinguish between 'AB' and 'BA'). If specified and a character, it must be in quotes.
BB	the character or number corresponding to the 'BB' snp genotype in the marker genotype file. This need only be specified if the file type is 'text'. If a character, then it must be in quotes.

availmemGb	a numeric value. It specifies the amount of available memory (in Gigabytes). This should be set to be as large as possible for best performance.
quiet	a logical value. If set to TRUE, additional runtime output is printed.

Details

ReadMarker can handle three different types of marker data; namely, genotype data in a plain text file, PLINK ped files, and vcf files.

Reading in a plain text file containing the marker genotypes: To load a text file that contains snp genotypes, run ReadMarker with filename set to the name of the file, and AA, AB, BB set to the corresponding genotype values. The genotype values in the text file can be numeric, character, or a mix of both.

We make the following assumptions

- The text file does not contain row or column headings
- The file is allowed to contain missing genotypes that have been coded according to missing
- Individuals are diploid
- The rows of the text file are the individuals and the columns are the marker loci
- The file is space separated
- The mapping of the observed genotypes in the marker file to AA, AB, and BB, remains the same for all loci
- Individuals are outbred when AA, AB, and BB are specified and inbred when only AA, and BB are specified
- For a text file, the same alphanumeric value is used for all missing marker genotypes. For a PLINK ped file, the missing allele is allowed to be '0' or '-'.

For example, suppose we have a space separated text file with marker genotype data collected from five snp loci on three individuals where the snp genotype AA has been coded 0, the snp genotype AB has been coded 1, the snp genotype BB has been coded 2, and missing genotypes are coded as 99

0 1 2 0 2
1 1 0 2 0
2 2 1 1 99

The file is called geno.txt and is located in the directory /my/dir/.

To load these data, we would use the command

```
geno_obj <- ReadMarker(filename='/my/dir/geno.txt', AA=0, AB=1, BB=2, type='text', missing=99)
```

where the results from running the function are placed in geno_obj.

As another example, suppose we have a space separated text file with marker genotype data collected from five snp loci on three individuals where the snp genotype AA has been coded a/a, the snp genotype AB has been coded a/b, and the snp genotype BB has been coded b/b

a/a a/b b/b a/a b/b
a/b a/b a/a b/b a/a
b/b b/b a/b a/b NA

The file is called geno.txt and is located in the same directory from which R is being run (i.e. the working directory).

To load these data, we would use the command

```
geno_obj <- ReadMarker(filename='geno.txt', AA='a/a', AB='a/b', BB='b/b',
                                        type='text', missing = 'NA')
```

where the results from running the function are placed in geno_obj.

Reading in a PLINK ped file: PLINK is a well known toolkit for the analysis of genome-wide association data. See https://www.cog-genomics.org/plink2 for details.

Full details of PLINK ped files can be found https://www.cog-genomics.org/plink/1.9/formats#ped. Briefly, the PED file is a space delimited file (tabs are not allowed): the first six columns are mandatory:

Family ID
Individual ID
Paternal ID
Maternal ID
Sex (1=male; 2=female; other=unknown)
Phenotype

Here, these columns can be any values since ReadMarker ignores these columns.

Genotypes (column 7 onwards) can be any character (e.g. 1,2,3,4 or A,C,G,T or anything else) except 0 which is, by default, the missing genotype character. All markers should be biallelic. All snps must have two alleles specified. Missing alleles (i.e 0 or -) are allowed. No column headings should be given.

As an example, suppose we have data on three individuals genotyped for four snp loci

FAM001	101	0	0	1	0	A	G	C	C	C	G	A	A
FAM001	201	0	0	2	0	A	A	C	T	G	G	T	A
FAM001	300	101	201	2	0	G	A	T	T	C	G	A	T

Then to load these data, we would use the command

```
geno_obj <- ReadMarker(filename='PLINK.ped', type='PLINK')
```

where geno_obj is used by AM, and the file PLINK.ped is located in the working directory (i.e. the directory from which R is being run).

Reading in a vcf file: VCF is a tab separated text file containing meta-information lines, a header line, and data lines. The data lines contain information about a position in the genome.

It is assumed that genotype information has been recorded on samples for each position.

Loci with more than two alleles will be removed automatically.

Eagle will only accept a single (uncompressed) vcf file. If chromosomal information has been recorded in separate vcf files, these files need to be merged into a single vcf file. This can be done by using the BCFtools utility set with command line "bcftools concat".

Value

To allow Eagle to handle data larger than the memory capacity of a machine, ReadMarker doesn't load the marker data into memory. Instead, it writes a reformatted version of the marker data, and its transpose, to the harddrive. These two files are only temporary, being removed at the end of the R session. The object returned by ReadMarker is a list object with the elements tmpM , tmpMt, and dim_of_M which is the full file name (name and path) of the reformatted file for the marker data, the full file name of the reformatted file for the transpose of the marker data, and a 2 element vector with the first element the number of individuals and the second element the number of marker loci.

Examples

```
#---------------------------------
#  Example 1
#---------------------------------
#
# Read in the genotype data contained in the text file geno.txt
#
# The function system.file() gives the full file name (name + full path).
complete.name <- system.file('extdata', 'geno.txt', package='Eagle')
#
# The full path and name of the file is
print(complete.name)

# Here, 0 values are being treated as genotype AA,
# 1 values are being treated as genotype AB,
# and 2 values are being treated as genotype BB.
# 4 gigabytes of memory has been specified.
# The file is space separated with the rows the individuals
# and the columns the snp loci.
geno_obj <- ReadMarker(filename=complete.name, type='text', AA=0, AB=1, BB=2, availmemGb=4)

# view list contents of geno_obj
print(geno_obj)

#---------------------------------
#  Example 2
#---------------------------------
#
# Read in the allelic data contained in the PLINK ped file geno.ped
#
# The function system.file() gives the full file name (name + full path).
complete.name <- system.file('extdata', 'geno.ped', package='Eagle')

#
# The full path and name of the file is
print(complete.name)

# Here,  the first 6 columns are being ignored and the allelic
# information in columns 7 -  10002 is being converted into a reformatted file.
# 4 gigabytes of memory has been specified.
# The file is space separated with the rows the individuals
```

```
# and the columns the snp loci.
geno_obj <- ReadMarker(filename=complete.name, type='PLINK', availmemGb=4)

# view list contents of geno_obj
print(geno_obj)



#--------------------------------
#  Example 3
#--------------------------------
#
#
# Read in the genotype data contained in the vcf file geno.vcf
#
# The function system.file() gives the full file name (name + full path).
complete.name <- system.file('extdata', 'geno.vcf', package='Eagle')
#
# The full path and name of the file is
print(complete.name)

# The file contains 5 marker loci recorded on 3 individuals
# Two of the loci contain multiple alleles and are removed.
# A summary of the file is printed once the file has been read.
geno_obj <- ReadMarker(filename=complete.name, type="vcf", availmemGb=4)

# view list contents of geno_obj
print(geno_obj)
```

ReadPheno *Read phenotype file*

Description

Read in the phenotype data.

Usage

```
ReadPheno(filename = NULL, header = TRUE, csv = FALSE, missing = "NA",
  ...)
```

Arguments

filename	contains the name of the phenotype file. The file name needs to be in quotes. If the file is not in the working directory, then the full path to the file is required.
header	a logical value. When TRUE, the first row of the file contains the names of the columns. Default is TRUE.

csv	a logical value. When TRUE, a csv file format is assumed. When FALSE, a space separated format is assumed. Default is FALSE.
missing	the number or character for a missing phenotype value.
...	arguments to be passed to read.table such as skip, sep. See read.table so the list of arguments.

Details

ReadPheno reads in the phenotype data which are data measured on traits and any fixed effects (or predictors/features/explanatory variables). A space separated plain text file is assumed. Each row in this file corresponds to an individual. The number of rows in the phenotype file must be the same as the number of rows in the marker data file. Also, the ordering of the individuals must be the same in the two files. A space separated file with column headings is the default but can be changed with the header and csv options.

The phenotype file may contain multiple traits and fixed effects variables.

Missing values are allowed. Eagle is told which value should be treated as missing by setting the missing parameter to the value.

For example, suppose we have three individuals for which we have collected data on two quantitative traits (y1 and y2), and four explanatory variables (age, weight, height, and sex). The data looks like

y1	y2	age	weight	height	sex
112.02	-3.123	26	75	168.5	M
156.44	1.2	45	102	NA	NA
10.3	NA	28	98	189.4	F

where the first row has the column headings and the next three rows contain the observed data on three individuals.

To load these data, we would use the command

```
pheno_obj <- ReadPheno(filename='pheno.dat', missing='NA')
```

where pheno.dat is the name of the phenotype file, and pheno_obj is the R object that contains the results from reading in the phenotype data. The file is located in the working directory so there is no need to specify the full path, just the file name is suffice.

Dealing with missing trait data:

AM deals automatically with individuals with missing trait data. These individuals are removed from the analysis and a warning message is generated.

Dealing with missing fixed effects values:

AM deals automatically with individuals with missing fixed effects values. These individuals are removed from the analysis and a warning message is generated

Value

a data frame is returned of the phenotype data. If `header` is true, the names of the columns will be as specified by the first row of the phenotype file. If `header` is `FALSE`, generic names are supplied by R in the form of V1, V2, etc. If no column headings are given, these generic names will need to be used in the `trait` and `fformula` parameters in `AM`. You can print out the column names of the data frame by using

```
names(pheno_obj)
```

The column names are also printed along with other summary information when `ReadPheno` is run.

See Also

`ReadMarker` for reading in marker data, `AM` for performing association mapping.

Examples

```
# Read in  phenotype data from ./extdata/

# find the full location of the phenotype data
complete.name <- system.file('extdata', 'pheno.txt', package='Eagle')

pheno_obj <- ReadPheno(filename=complete.name)

 ## print a couple of lines of the data file
 head(pheno_obj)
```

ReadZmat *Read Z matrix*

Description

Read in the Z matrix that assigns groups/strains/lines to their trait measurements.

Usage

```
ReadZmat(filename = NULL)
```

Arguments

`filename`	contains the name of the Z matrix file. The file name needs to be in quotes. If the file is not in the working directory, then the full path to the file is required.

Details

The underlying linear mixed model is of the form

$$Y = X\beta + Zu_g + e$$

where Z is a (n x n_g) matrix that contains ones and zeros, n is the number of trait measurements, and n_g is the number of groups/strains/lines. If n and n_g are the same, then there is no need to specify Z. However, if a group/strain/line has multiple trait measurements (i.e. $n > n_g$) then the Z matrix is needed to tell Eagle which trait measurements belong to which groups/strains/lines.

A space separated text file is assumed. Each row of the matrix contains multiple zeroes but only a single one. The file cannot contain column or row headings. The file also cannot contain a row of only zeroes. Here, n must be larger than n_g otherwise an error will be issued.

Value

a data matrix is returned of the Z matrix.

See Also

ReadMarker and ReadPheno.

Examples

```
# Read in  example Z matrix from ./extdata/

# find the full location of the Z matrix data
complete.name <- system.file('extdata', 'Z.txt', package='Eagle')

# read in Z matrix data
Z_obj <- ReadZmat(filename=complete.name)

# look at first few rows of the Z matrix file
head(Z_obj)
```

SummaryAM *Summary of multiple locus association mapping results*

Description

A summary function that provides additional information on the significant marker-trait associations found by AM

Usage

```
SummaryAM(AMobj = NULL)
```

Arguments

AMobj	the (list) object obtained from running AM.

Details

SummaryAM produces two tables, an overall summary table and a table of results with the p-value for each fixed effect in the final model.

See Also

AM

Examples

```
## Not run:
 # Since the following code takes longer than 5 seconds to run, it has been tagged as dontrun.
  # However, the code can be run by the user.
  #

  #---------------
  # read the map
  #---------------
  #
  # File is a plain space separated text file with the first row
  # the column headings
  complete.name <- system.file('extdata', 'map.txt',
                                  package='Eagle')
  map_obj <- ReadMap(filename=complete.name)

 # to look at the first few rows of the map file
 head(map_obj)

  #------------------
  # read marker data
  #------------------
  # Reading in a PLINK ped file
 # and setting the available memory on the machine for the reading of the data to 8 gigabytes
  complete.name <- system.file('extdata', 'geno.ped',
                                    package='Eagle')
  geno_obj <- ReadMarker(filename=complete.name,  type='PLINK', availmemGb=8)

  #----------------------
  # read phenotype data
  #----------------------

  # Read in a plain text file with data on a single trait and two fixed effects
  # The first row of the text file contains the column names y, cov1, and cov2.
  complete.name <- system.file('extdata', 'pheno.txt', package='Eagle')

  pheno_obj <- ReadPheno(filename=complete.name)

  #-------------------------------------------------------
```

```
  # Perform multiple-locus genome-wide association mapping
  #-------------------------------------------------------
  res <- AM(trait = 'y',
                           fformula=c("cov1 + cov2"),
                           map = map_obj,
                           pheno = pheno_obj,
                           geno = geno_obj)

  #----------------------------------------
  # Produce additional summary information
  #----------------------------------------

  SummaryAM(AMobj=res)

## End(Not run)
```

Index