Package 'Eagle'

May 4, 2017

Type Package

Title Multiple Locus Association Mapping on a Genome-Wide Scale

Version 1.0.0

Date 2017-05-17

Author Andrew W. George, Joshua Bowden, Ryan Stephenson

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

Description An implementation of multiple-locus association mapping on a genome-wide scale. Eagle can handle inbred or 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, linear mixed models can tolerate non-normal data. Eagle reports, as its findings, the best set of snp in strongest association with a trait. To perform an analysis, run OpenUI(). 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 (>= 3.3.2)

Suggests knitr, rmarkdown

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

LinkingTo RcppEigen, Rcpp

Roxygen list(wrap=FALSE)

LazyData true

VignetteBuilder knitr

NeedsCompilation yes

R topics documented:

`Eagle-package` *Multiple-locus Genome-Wide Association Mapping*

Description

A package for making genome-wide association mapping with multiple-locus linear mixed models routine.

Details

Package documentation

Motivation

Data from genome-wide association studies are analysed, commonly, with single-locus models. That is, analyses 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. Much effort has been devoted to making the package as easy to use as possible. As part of this effort, we developed a web-based user interface to Eagle, sheilding users from having to write R code to run the functions.

Assumptions

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

Reading in Marker Data

Our package can accept genotype data in plain text files and allelic data in PLINK ped files. These files can be larger than the memory capacity of the machine. Other formats can also be handled via the PLINK package and using the `recode` option to convert differnt format marker files into ped files. See `ReadMarker` for more details.

Quick start guide

At the R prompt, run

```
OpenUI()
```

This opens a web browser to the user interface for Eagle. Here, you can read in the data, analyse the data, and summarize the findings.

Output

The aim of a genome-wide association study (GWAS) is to identify those marker loci closest to the genes that are influencing the trait. So, when the GWAS data are analysed, a set of marker loci labels are returned as the output. These marker loci are closest to the genes underlying the trait and are found while simultaneously accounting for multiple marker-trait associations, familial relatedness, and fixed effects such as population structure (if included in the analysis). More detailed output such as the additive effect of the marker locus, its significance in the multiple-locus model (measured by a p-value), and an estimate of the amount of variation explained by the locus can be obtained by running the summary function `SummaryAM`.

Where to get help

A variety of different help options are available.

- At the R propmpt, type

  `library(, "Eagle")`

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

  `help("nameoffunction")`
- A QuickStart vignette and Example vignette are available by typing, at the R prompt,

  `vignette("QuickStart", "Eagle")`

  and

  `vignette("Example", "Eagle")`

  , respectively.
- Email <Eagle@csiro.au>

Author(s)

Andrew W. George <andrew.george@csiro.au> with lots of help from Joshua Bowden (CSIRO).

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, availmemGb = 8, geno = NULL,
  pheno = NULL, map = NULL, ncpu = detectCores(), ngpu = 0,
  quiet = TRUE, maxit = 20)
```

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.
availmemGb	a numeric value. It specifies the amount of available memory (in Gigabytes). This should be set to the maximum practical value of available memory for the analysis.
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 specifed, a generic map 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 TRUE, 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.

Details

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 numeric value 9 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". The file is located in the current working directory.
- there is no map data.

To analyse these data, we would use the following functions:

```
geno_obj <-  ReadMarker(filename="geno.txt", AA=0, AB=1, BB=2, type="text", missing=9)

pheno_obj <- ReadPheno(filename="pheno.txt", header=TRUE)

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:

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 Gbytes, and our computer only has 32 Gbytes 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 labelled "y1". The second column is another trait and is laballed "y2". The third and fourth columns are nuisance variables and are labelled "cov1" and "cov2". The fifth and sixth columns are the first two principal components to account for population substructure and are labelled "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", the fixed effects part of the model is "cov1 + cov2 + pc1 + pc2", and the available memory is set to 32 Gbytes.

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", header=TRUE, missing=99)

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

res <- AM(trait="y2", fformula=c("cov1 + cov2 + pc1 + pc2"),
          geno=geno_obj, pheno=pheno_obj, map=map_obj, availmemGb=32)
```

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, MACH, fastPHASE, 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=TRUE`, 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 Right hand size formula of the fixed effects part of the linear mixed model.

indxNA a vector containing the row indexes of those individuals, whose trait and fixed effects data contain missing values and have been removed from the analysis.

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 Gbytes 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.

See Also

`ReadMarker`, `ReadPheno`, and `ReadMap`

`OpenUI` *Browser-based User Interface*

Description

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

Usage

```
OpenUI()
```

Details

`OpenUI` is an easy to use web-based user interface for Eagle. By clicking on the navigation tabs at the top of a page, data can be read and analysed. By using this user interface, a user can 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
OpenUI()

## End(Not run)
```

ReadMap *Read map file*

Description

Read in the marker map data. The first row is assumed to contain the column headings.

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.
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 assmed 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 not allowed.

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

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 marker data. Three different types of data can be read.

Usage

```
ReadMarker(filename = NULL, type = "text", missing = NULL, AA = NULL,
  AB = NULL, BB = NULL, availmemGb = 8, quiet = TRUE)
```

Arguments

`filename`	contains the name of the marker file. The file name needs to be in quotes.
`type`	specify the type of file. Choices are "text" and "PLINK".
`missing`	the number or character for a missing genotype in the text file. There is no need to specify this for a PLINK ped file. Missing allele values in a PLINK file must be coded as "0" or "-".
`AA`	the character(s) or number corresponding to the AA snp genotype in the marker genotype file. This must be specified if the file type is "text". The character(s) must be in quotes.
`AB`	the character(s) or number corresponding to the AB snp genotype in the marker genotype file. This can be left unspecified if there are no heterozygote genotypes (i.e. the individuals are inbred). If specified, the character(s) must be in quotes.
`BB`	the character(s) or number corresponding to the BB snp genotype in the marker genotype file. This must be specified if the file type is "text". The character(s) 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 along with additional error checking. Specifically, `ReadMarker` will check that the marker file has the same number of entries per row.

Details

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

Reading in previously read marker data: To read marker data that has been previously read with ReadMarker in another R session, run the function with no arguments. For example

```
geno_obj <- ReadMarker()
```

where geno_obj is the name of the user defined R object that is to contain the marker data.

For this command to work without error, the working directory needs to be the same as the working directory form which the ReadMarker was first run. That is, R needs to be started from the same directory for both sessions. You can check what the current working directory is with the command

```
getwd()
```

Loading the marker data in this way is much faster than reading the data again from file because there is no need to pre-process the data, nor check the data for errors.

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 "-".

As an 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/".

Then 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.

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).

Then 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 =
```

where `geno_obj` is used by `AM`.

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 other formats: Having first installed the stand-alone PLINK software, it is possible to convert other file formats into PLINK ped files. See https://www.cog-genomics.org/plink/1.9/formats for details.

For example, to convert vcf file into a PLINK ped file, at the unix prompt, use the PLINK command

```
PLINK --vcf filename.vcf --recode --out newfilename
```

and to convert a binary ped file (bed) into a ped file, use the PLINK command

```
PLINK --bfile filename --recode --tab --out newfilename
```

Value

To allow AM to handle data larger than the memory capacity of a machine, ReadMarker doesn't load the marker data into memory. Instead, it creates a reformatted file of the marker data and its transpose. The object returned by ReadMarker is a list object with the elements asciifileM , asciifileMt, and dim_of_ascii_M which is the full file name (name and path) of the refomatted 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")


# Here,  0 values are being treated as genotype AA,
# 1 values are being treated as genoytpe AB,
# and 2 values are being treated as genoytpe BB.
# 4 Gbytes 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")


# Here,  the first 6 columns are being ignored and the allelic
# information in columns 7 -  10002 is being converted into a packed binary file.
# 4 Gbytes 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)
```

ReadPheno *Read phenotype file*

Description

Read in the phenotype data.

Usage

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

Arguments

filename	contains the name of the phenotype file. The file name needs to be in quotes.
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.
...	any other read.table formatting commands

Details

ReadPheno reads in the phenotype data. 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.

As an 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

Then to load these data, we would use the command

```
pheno_obj <- ReadPhenr(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 and file name.

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 dataframe by using

```
names(pheno_obj)
```

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)
```

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, pheno = NULL, geno = NULL, map = NULL)
```

Arguments

AMobj	the (list) object obtained from running AM. Must be specified.
pheno	the (data frame) object obtained from running ReadPheno. Must be specified.
geno	the (list) object obtained from running ReadMarker. Must be specified.
map	the (data frame) object obtained from running ReadMap. The default is to assume a map object has not been supplied. Optional.

Details

SummaryAM produces two tables of results. First, a table of results is produced with the additive effect size and p-value for each fixed effect in the final model. Second, a table of results is produced with the proportion of phenotypic variance explained by the different multiple-locus models. Each row in this table is the proportion of phenotypic variance after the marker locus has been added to the multiple locus model. Our calculations of variance explained are based on Sun et al. (2010).

References

Sun G., Zhu C., Kramer MH., Yang S-S., et al. 2010. Variation explained in mixed model association mapping. Heredity 105, 330-340.

See Also

AM

Examples

```
 #---------------
 # 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 8Gbytes
 complete.name <- system.file("extdata", "geno.ped",
                                   package="Eagle")
 geno_obj <- ReadMarker(filename=complete.name,  type="PLINK", availmemGb=8)

 #----------------------
 # read phenotypic 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, availmemGb=8)

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

SummaryAM(AMobj=res, pheno=pheno_obj, geno=geno_obj, map=map_obj)
```

Index