Package ‘gdmp’

March 29, 2019

Type Package

Title Genomic Data Management

Version 0.2.0

Date 2019-03-29

Maintainer Gamal Abdel-Azim <gamal.azim@gmail.com>

Description Manage and analyze high-dimensional SNP data from chips with multiple densities.

Depends methods, R(>= 3.0.0)

License GPL (>= 2)

LazyData TRUE

RoxygenNote 5.0.1

NeedsCompilation yes

Author Gamal Abdel-Azim [aut, cre]

Repository CRAN

Date/Publication 2019-03-29 17:00:02 UTC

R topics documented:

arrayAppend *Append two arrays by adding new data to top array*

Description

`arrayAppend` is a function to append two arrays by adding new data to array whose name is placed in the first argument of the function.

Usage

```
arrayAppend(topA, bottomA, missingVal = 5)
```

Arguments

`topA`	Old data.
`bottomA`	New data.
`missingVal`	Value to use for missing SNPs in new data, default = 5.

Details

Two matrices created by the function, `toArray`, can be appended together using `arrayAppend`. Appending is more efficient after recoding each of the two matrices by using `snpRecode`. The first input argument `topA` holds old data, i.e., data to hold on to its SNP list. The second argument, `bottomA`, holds new data that needs to be fit to old SNP list. Therefore, missing SNPs in new data will be created anew but will be given the value of 'missingVal', with a default of 5. This also assumes that you will be imputing these missing values later on.

`arrayAppend` goes by the SNP list of the old data in the first input argument, so if the new data matrix has more SNPs, they will be ignored. If you need to consider the extra SNPs, switch the order of the input matrices of the function.

Value

An object of class matrix with 'number of rows = number of individuals in both matrices, topA and bottomA' and 'number of columns = number of SNPs as listed in old data'.

See Also

`toArray`

Examples

```
ga.old <- matrix(sample(c(0,1,2,5), size = 30, repl = TRUE), nrow=3, ncol=10,
  dimnames = list(paste("Individual", 1:3, sep="."), paste("SNP", 1:10, sep=".")))

ga <- matrix(sample(c(0,1,2,5), size = 24, repl = TRUE), nrow=3, ncol=8,
dimnames = list(paste("Individual", 4:6, sep="."), paste("SNP", 1:8, sep=".")))
```

```
arrayAppend(ga.old, ga)
arrayAppend(ga.old, ga)

#
#             SNP.1 SNP.2 SNP.3 SNP.4 SNP.5 SNP.6 SNP.7 SNP.8 SNP.9 SNP.10
#Individual.1     5     5     0     1     2     1     2     5     1      2
#Individual.2     2     2     2     0     1     1     1     1     2      0
#Individual.3     1     1     5     0     5     5     0     5     5      5
#Individual.4     1     2     1     1     0     2     0     0     5      5
#Individual.5     2     1     2     1     0     1     2     1     5      5
#Individual.6     0     2     0     2     5     0     1     1     5      5
#
#Note that SNP.9 and SNP.10 were added to new data but were assigned the default
#code, 5, for missing values. You should impute all missing values afterwards.
#

#If the order of input matrices is switched, the extra SNPs in the second argument
#will be ignored. This function goes by the SNP list of the first arguement.

arrayAppend(ga, ga.old)

#             SNP.1 SNP.2 SNP.3 SNP.4 SNP.5 SNP.6 SNP.7 SNP.8
#Individual.4     1     2     1     1     0     2     0     0
#Individual.5     2     1     2     1     0     1     2     1
#Individual.6     0     2     0     2     5     0     1     1
#Individual.1     5     5     0     1     2     1     2     5
#Individual.2     2     2     2     0     1     1     1     1
#Individual.3     1     1     5     0     5     5     0     5
```

`BayesAB` *Estimte SNP Effects by MCMC - Bayes (A and B) - beta*

Description

`BayesAB` is an implementation of Bayes A and B for estimating direct SNP effects in high dimensional data problems ($p » N$). `BayesAB` utilizes the C Function cBaysABF for speed ..

Usage

```
BayesAB(ga, numiter = 5000, numMHIter = 10, Pi = .9, y)
```

Arguments

`ga`	A matrix of genotypes with a number of rows identical to the number of genotyped individuals and a number of columns identical to the number of SNPs. Values in the matrix are 0, 1, 2, & 5 for homozygous, heterozygous, other homozygous, & unknown genotypes, respectively.

`numiter`	Number of iterations
`numMHIter`	Number of Metropolis-Hastings iterations. Set to a value of 1 with Bayes A.
`Pi`	Proportion of SNP loci with 0 effect. Set to 0 to run Bayes A instead of B.
`y`	Trait phenotypes or conventional breeding values

Details

This function runs Bayes A and B to estimate direct SNP effects based on a matrix of genotypes, `ga` and a vector of adjusted phenotypes, `y`, (Meuwissen et al., 2001; Genetics 157:1819-1829). This is essential in high dimensional data problems with highly overparameterized models (*p » N*). To run Bayes A, set `Pi` to 0 and `numMHIter` to 1.

Other data management functions in `gdmp` can be used to construct the integer matrix of genotypes, `ga`, to use as input to `BayesAB`.

Value

A list object with a vector of SNP estimates, `meanb`, and a vector of genomic values for individuals, `aHat`, are returned. It is also possible to extract the number of SNP loci in `nLoci`.

References

Meuwissen et al. (2001). Prediction of Total Genetic Value Using Genome-Wide Dense Marker Maps. *Genetics*, **157**, 1819–1829.

See Also

`BayesCpi`

`BayesCpi` *Estimte SNP Effects by MCMC - Bayes C and Cpi - beta*

Description

`BayesCpi` is an implementation of Bayes Cpi to extend Bayes A and B for estimating direct SNP effects in high dimensional data problems (*p » N*). `BayesCpi` treats the prior probability, `pi` = *P(SNP has zero effect)*, as unknown. The C Function `cBaysCpi` is utilized for for speed ..

Usage

```
BayesCpi(ga, numiter = 5000, Pi = .9, y)
```

Arguments

`ga`	A matrix of genotypes with a number of rows identical to the number of genotyped individuals and a number of columns identical to the number of SNPs. Values in the matrix are 0, 1, 2, & 5 for homozygous, heterozygous, other homozygous, & unknown genotypes, respectively.
`numiter`	Number of iterations
`Pi`	Proportion of SNP loci with 0 effect for Bayes C
`y`	Trait phenotypes or conventional breeding values

Details

This function runs Bayes C and Cpi to estimate direct SNP effects and the proportion of loci with nonzero effects based on a matrix of genotypes, `ga` and a vector of adjusted phenotypes, `y`, (Habier et al., 2011; BMC Bioinformatics 12:186). As in other bayesian alphabet, Bayes Cpi is essential in high dimensional data problems with highly overparameterized models (p » N). It extends Bayes A and B to estimate the proportion of loci with nonzero effect.

Other data management functions in `gdmp` can be used to construct the integer matrix of genotypes, `ga`, to use as input to `BayesCpi`.

Value

A list object with a vector of SNP estimates `meanb` and a vector of genomic values for individuals, `aHat` are returned. It is also possible to extract the estimated number of SNP loci in `nLoci`.

References

Habier et al. (2011). Extension of the bayesian alphabet for genomic selection. *BMC Bioinformatics*, **12**, 186.

See Also

`BayesAB`

`GetHCS` *Check SNP genotypes and exclude redundant SNPs*

Description

Given individual genotypes of a set of SNPs, the function checks for the existence of redundant SNPs to exclude. A margin of error of 0.5% is allowed by default.

Usage

```
GetHCS(ga.r, Exclude=1:ncol(ga.r), allow = .005)
```

Arguments

`ga.r`	Matrix of genotypes created by `toArray` and converted to integer genotypes matrix by `snpRecode`.
`Exclude`	Numeric indices or column names of `ga.r` to check.
`allow`	allowed margin of error, default is 0.005.

Details

Test for similar SNP genotypes across a set of individuals. SNPs are considered identical if the number of different genotypes in the population tested remains below an allowed error margin of 0.5%. Say, `Exclude <- 1:100` with SNP #1 similar to #25, then `Exclude[25]` will be flagged for exclusion, whereas `Exclude[1]` will not be flagged for exclusion.

In addition to identical SNPs, the function flaggs SNP genotypes that are entirely opposite within error margin as redundant as well. Thus, SNPs are declared highly correlated if the genotypes are all the same (0-0, 1-1, and 2-2) or all opposite (0-2, 1-1, 2-0) within the error margin specified.

Value

If `Exclude` contains SNP names, a character vector of excluded SNPs is returned, and if it contains integer values, a numeric vector of excluded SNPs is returned.

See Also

`is.identical`, `snpRecode`, `toArray`

Examples

```
## Simulate random allele designations for 100 bi-allelic SNPs
set.seed(2016)
desig <- array(sample(c('A','C','G','T'), size = 200, repl = TRUE), dim=c(100, 2))

## Simulate random SNP genotypes for 20 individuals - put them in array format
## '-' indicates an unknown base
ga <- array(0, dim=c(20, 100))
for(i in 1:20)
  for(j in 1:100)
   ga[i, j] <- paste(sample(c(desig[j,],"-"), 2, prob=c(.47, .47, .06), repl=TRUE), collapse='')

## Recode the matrix, place recoded genotypes in ga.r
desig <- data.frame(AlleleA_Forward = factor(desig[,1]), AlleleB_Forward = factor(desig[,2]))
ga.r <- array(5, dim=c(20, 100))
for(i in 1:100) ga.r[,i] <- snpRecode(ga[,i], desig[i,])

## Check all SNP genotypes in ga.r for similarity across individuals
## Allow for a margin of error of 0.5%
GetHCS(ga.r)
#[1] 42 91  # SNPs 42 & 91 are similar to earlier SNPs in the vector, 'Exclude'

## Check SNP genotypes from 1 to 50 for similarity across individuals
```

```
GetHCS(ga.r, Exclude=1:50)
#[1] 42
```

getMAF *Return frequency of the minor allele of a SNP*

Description

Given individual genotypes of a single SNP, the function returns allele frequency of the minor allele.

Usage

```
getMAF(snpG)
```

Arguments

snpG	is a column vector in the genotypes array, created by `toArray` and converted to integer genotypes by `snpRecode`. The column represents genotypes of a single SNP for all individuals in data.

Details

Allele frequency is calculated for one of the two alleles and is returned if it is below 0.5, otherwise '1 - allele frequency' is returned. The function retuns a frequency of 0 for non polymorphic SNPs.

Value

Allele frequency of the minor allele.

References

Falconer and Mackay (1996). Introduction to Quantitative Genetics (4th Edition). *Pearson Education Limited, Edinburgh, England.*

See Also

`snpSelect`, `snpRecode`, `toArray`

Examples

```
set.seed(002016)
snpG.1 <- c(rep(0, 100), rep(1, 80), rep(2, 9))
snpG.2 <- c(rep(0, 100), rep(1, 80))
snpG.3 <- c(rep(0, 100), rep(2, 9))
snpG.4 <- c(rep(0, 100))

getMAF(snpG.1)
#       0
```

```
#0.2592593

getMAF(snpG.2)
#       1
#0.2222222

getMAF(snpG.3)
#        0
#0.08256881

getMAF(snpG.3)
```

is.hwEq *Check a SNP for Hardy-Weinberg equilibrium*

Description

Given individual genotypes of a single SNP, the function checks a population of individuals for Hardy-Weinberg equilibrium.

Usage

```
is.hwEq(snpG, diff)
```

Arguments

`snpG`	is a column vector in the genotypes array, created by `toArray` and converted to integer genotypes by `snpRecode`. The column represents genotypes of a single SNP for all individuals in data.
`diff`	heterozygosity difference used in HW equilibrium, see 'Details'.

Details

A logical function to check for HW equilibrium, the expected frequency of the heterozygous genotype is estimated based on the two homozygous genetypes by 'sqrt(4*p^2*q^2)'. The absolute difference between the expected and observed frequency of the heterozygous genotype of the SNP needs to be smaller than the minimum difference of `diff`.

Value

A logical TRUE for H-W equilibrium or FALSE for H-W disequilibrium is returned.

References

Falconer and Mackay (1996). Introduction to Quantitative Genetics (4th Edition). *Pearson Education Limited, Edinburgh, England.*

Wiggans et al. (2009). Selection of single-nucleotide polymorphisms and quality of genotypes used in genomic evaluation of dairy cattle in the United States and Canada. *Journal of Dairy Science*, **92**, 3431-3436.

See Also

snpSelect, snpRecode, toArray

Examples

```
## Simulate random allele designations for 100 bi-allelic SNPs
set.seed(2016)
desig <- array(sample(c('A','C','G','T'), size = 200, repl = TRUE), dim=c(100, 2))

## Simulate random SNP genotypes for 20 individuals - put them in array format
## '-' indicates an unknown base
ga <- array(0, dim=c(20, 100))
for(i in 1:20)
  for(j in 1:100)
    ga[i, j] <- paste(sample(c(desig[j,],"-"), 2, prob=c(.47, .47, .06), repl = TRUE), collapse='')

## Recode the matrix, place recoded genotypes in ga.r
desig <- data.frame(AlleleA_Forward = factor(desig[,1]), AlleleB_Forward = factor(desig[,2]))
ga.r <- array(5, dim=c(20, 100))
for(i in 1:100) ga.r[,i] <- snpRecode(ga[,i], desig[i,])

## Check the first 10 SNPs for H-W equilibrium based on a minimum
## allowed difference of 0.15 between observed and expected heterozygosity
apply(ga.r[,1:10], 2, is.hwEq, diff=.15)
# [1]  TRUE FALSE FALSE  TRUE  TRUE FALSE FALSE FALSE  TRUE FALSE
```

---

is.identical *Check two SNPs for near identity across genotyped samples*

Description

Given individual genotypes of two SNPs, the function checks if the two sets of genotypes are identical or completely opposite. A margin of error of 0.5% is allowed.

Usage

```
is.identical(x, y, allow = .005)
```

Arguments

x, y	two column vectors in the genotypes array, created by toArray and converted to integer genotypes by snpRecode. The two columns are genotypes of two SNPs for all individuals in data.
allow	allowed margin of error

Details

Test if two SNPs are identical in genotypes across a set of individuals. SNPs are considered identical if the number of different genotypes in the population tested remains below an allowed error margin of 0.5%.

In addition to identical SNPs, the function considers SNP genotypes that are entirely opposite within error margin as redundant. Thus, two SNPs are declared highly correlated if the genotypes are all the same (0-0, 1-1, and 2-2) or are all opposite (0-2, 1-1, 2-0) within the error margin specified.

Value

A logical value, of TRUE for identical SNPs or FALSE for different SNPs, is returned.

See Also

`GetHCS`, `snpRecode`, `toArray`

Examples

```
## Simulate random allele designations for 100 bi-allelic SNPs
set.seed(2016)
desig <- array(sample(c('A','C','G','T'), size = 200, repl = TRUE), dim=c(100, 2))

## Simulate random SNP genotypes for 20 individuals - put them in array format
## '-' indicates an unknown base
ga <- array(0, dim=c(20, 100))
for(i in 1:20)
  for(j in 1:100)
   ga[i, j] <- paste(sample(c(desig[j,],"-"), 2, prob=c(.47, .47, .06), repl=TRUE), collapse='')

## Recode the matrix, place recoded genotypes in ga.r
desig <- data.frame(AlleleA_Forward = factor(desig[,1]), AlleleB_Forward = factor(desig[,2]))
ga.r <- array(5, dim=c(20, 100))
for(i in 1:100) ga.r[,i] <- snpRecode(ga[,i], desig[i,])

## Check the first 2 SNPs for being identical based
## on a minimum allowed margin of error of 0.5%
is.identical(ga.r[,1], ga.r[,2], allow = .005)
# [1] FALSE

## Create an instance of exactly identical SNP genotypes
ga.r <- cbind(ga.r, ga.r[,1])  # SNP #1 and #101 are exactly identical
is.identical(ga.r[,1], ga.r[,101], allow = 0)
# [1] TRUE

## Create an instance of identical SNP genotypes with a 5% error
ga.r <- cbind(ga.r, ga.r[,1])  # SNP #1 and #101 are 100% identical
ga.r[20,101] <- 2  # a different genotype, to make SNP #1 & #101 only 95% identical
is.identical(ga.r[,1], ga.r[,101]) # use default allow of .005
# [1] FALSE
```

```
is.identical(ga.r[,1], ga.r[,101], allow = .05) # allow for a 5% marging of error
# [1] TRUE
```

read.findhap *Read the output of Paul Vanraden's findhap, 'genotypes.filled'*

Description

`read.findhap` is a tool that utilizes external C code to read quickly the output of findhap program stored by default in the 'genotypes.filled' file. 'findhap' is a standalone software program written by Paul Vanraden to impute missing genotypes. If you are not familiar with 'findhap', you probably don't need this function.

Usage

```
read.findhap(Nanim, Nmark, file="./genotypes.filled")
```

Arguments

`Nanim`	Total number of individuals to read from imputed genotypes file.
`Nmark`	Total number of SNP genotypes in the imputed-to chip.
`file`	Findhap's output file for imputed genotypes. Default is 'genotypes.filled'.

Details

This function reads into R environment imputed SNP genotypes as formated by findhap program. This tool utilizes external C code for quick read. The standard output of this function is an object of class matrix.

Value

An object of class matrix with 'number of rows = Nanim' and 'number of columns = Nmark + 3'. Refer to findhap manual for more details about the structure of 'genotypes.filled'.

References

Paul Vanraden. Find haplotypes and impute genotypes using multiple chip sets and sequence data. http://aipl.arsusda.gov/software/findhap/.

snpRecode *Recode a matrix of SNP genotypes as 0, 1, and 2*

Description

snpRecode is a function to convert SNP genotypes to 0, 1, and 2 for the homozygous, heterozygous, and other homozygous genotype, respectively.

Usage

```
snpRecode(snpG, designat)
```

Arguments

snpG	is a column vector in the genotypes array, created by toArray. The column represents genotypes of a single SNP for all or a subset of individuals in data.
designat	is the 2-base allele designations for each SNP. This is sometimes called allele report data, where the specefic bases of alleles A and B are reported. Formated as data frame with two factors for alleles A and B. See 'Examples'.

Details

Recode snp genotypes by counting the number of copies of allele A in an element of snpG which is a column vector in the genotypes array, ga, where

- snpG is a column vector in the genotypes array,
- ga is the genotypes array created by toArray. It contains elements such as "AA", "AG", "GA", "-A", "- -".

Unknown genotypes are those with non A/G/C/T bases, those are coded as 5.

Value

A column vector of the integers 0, 1, and 2 is created based on the number of copies of allele A in each element of the supplied vector of genotypes. A value of 5 is used to indicate an unknown genotype.

See Also

toArray

Examples

```
## Simulate random allele designations for 100 bi-allelic SNPs
set.seed(2016)
desig <- array(sample(c('A','C','G','T'), size = 200, repl = TRUE), dim=c(100, 2))
```

```
## Simulate random SNP genotypes for 20 individuals - put them in array format
## '-' indicates an unknown base
ga <- array(0, dim=c(20, 100))
for(i in 1:20)
  for(j in 1:100)
   ga[i, j] <- paste(sample(c(desig[j,],"-"), 2, prob=c(.46, .46, .08), repl=TRUE), collapse='')

## Recode the matrix, place recoded genotypes in ga.r
desig <- data.frame(AlleleA_Forward = factor(desig[,1]), AlleleB_Forward = factor(desig[,2]))
ga.r <- array(5, dim=c(20, 100))
for(i in 1:100) ga.r[,i] <- snpRecode(ga[,i], desig[i,])

## Tabulate recoded genotypes in the matrix ga.r
table(ga.r)
#   0   1   2   5
# 326 632 701 341
```

---

snpSelect — *Select genotyped individuals and SNPs based on call rate, heterozygosity, HW equilibrium, or minor allele frequency.*

Description

`snpSelect` is a function to exclude individuals and SNPs that do not satisfy one of the following criteria,

- SNP genotypes are above the provided call rate threshold, default = 0.85
- SNP genotypes are above the provided heterozygosity rate, default = 0.01
- SNP loci are not far from Hardy-Weinberg equilibrium
- SNP genotypes are above the provided minimum for minor allele frequency, default = 0.005

Usage

```
snpSelect(ga.r, select.method=c("call.rate", "heterozygosity", "HW.Eq", "MAF"),
  call.rate.min = .85, hz.min = .01, hz.diff = .15, MAF.min = .005)
```

Arguments

`ga.r`	Matrix of SNP genotypes as prepared by `toArray` and recoded by `snpRecode`.
`select.method`	The SNP criterion to use in selecting individuals to keep, default is 'call.rate'.
`call.rate.min`	Call rate threshold provided or the minimum allowed call rate.
`hz.min`	Heterozygosity threshold provided, less polymorphic SNPs are excluded.
`hz.diff`	Heterozygosity difference used in HW equilibrium, see 'Details'.
`MAF.min`	Minor allele frequency threshold provided, SNPs with lower frequencies for the minor allele are excluded.

Details

This is a selection tool to extract high quality individuals and SNPs. Based on the selection criterion provided, `snpSelect` keeps individuals and SNPs that meet the criterion. Providing multiple selection criteria to `select.method` does not work. If a cleanup requires extraction of individuals and SNPs based on multiple criteria, the function may be used iteratively.

In the case of `select.method` = 'call.rate', individuals not SNPs are excluded. In all other cases, SNPs are excluded.

For selection based on HW equilibrium, the expected frequency of the heterozygous genotype is estimated based on the two homozygous genetypes by 'sqrt(4*p^2*q^2)'. The absolute difference between the expected and observed frequency of the heterozygous genotype of each SNP needs to be smaller than the minimum difference of `hz.diff`. The logical function `is.hwEq(x, diff)` with a TRUE or FALSE output can be used to test SNP genotypes in the integer vector, `x`, based of the difference, `diff`.

Value

A matrix with individuals exceeding the `call.rate.min` threshold or a matrix with SNPs exceeding `hz.min`. A minimum heterozygosity instead of 0 is recommended to leave room for genotying errors. The function can also be used to exclude SNPs not in Hardy-Weinberg equilibrium if the difference between expected and observed genotype frequencies is greater than `hz.min`. Finally, `snpSelect` may be used to exclude SNPs with low minor allele frequencies. If `MAF.min` is not supplied, the default of 0.5% is used.

References

Falconer and Mackay (1996). Introduction to Quantitative Genetics (4th Edition). *Pearson Education Limited, Edinburgh, England.*

Wiggans et al. (2009). Selection of single-nucleotide polymorphisms and quality of genotypes used in genomic evaluation of dairy cattle in the United States and Canada. *Journal of Dairy Science*, **92**, 3431-3436.

See Also

`getMAF`, `is.hwEq`, `snpRecode`, `toArray`

Examples

```
## Simulate random allele designations for 100 bi-allelic SNPs
set.seed(2016)
desig <- array(sample(c('A','C','G','T'), size = 200, repl = TRUE), dim=c(100, 2))

## Simulate random SNP genotypes for 20 individuals - put them in array format
## '-' indicates an unknown base
ga <- array(0, dim=c(20, 100))
for(i in 1:20)
  for(j in 1:100)
   ga[i, j] <- paste(sample(c(desig[j,],"-"), 2, prob=c(.47, .47, .06), repl=TRUE), collapse='')
```

```
## Recode the matrix, place recoded genotypes in ga.r
desig <- data.frame(AlleleA_Forward = factor(desig[,1]), AlleleB_Forward = factor(desig[,2]))
ga.r <- array(5, dim=c(20, 100))
for(i in 1:100) ga.r[,i] <- snpRecode(ga[,i], desig[i,])

## Exclude individuals with call rates below 85%
dim(snpSelect(ga.r, select.method = "call.rate", call.rate.min = 0.85))
#[1]  15 100

## Exclude SNPs with heterozygosity <= 1%
dim(snpSelect(ga.r, select.method = "heterozygosity", hz.min = 0.01))
#[1] 20 79

## If the difference between expected and observed genotype frequencies
## is > 0.15, exclude the SNP.
dim(snpSelect(ga.r, select.method = "HW.Eq", hz.diff = 0.15))
#[1] 20 29

## Select SNPs with minor allele frequencies greater than 0.5%
dim(snpSelect(ga.r, select.method = "MAF", MAF.min = 0.005))
#[1] 20 79
```

toArray *Turn Illumina's Genome Studio 'Final Report' file into an array*

Description

`toArray` is a function to turn genotyping data into an array. This function does not read stored data in the 'Final Report' file, however, it converts a data fram with the same structure as 'Final Report' into an object of class matrix. The specifications of the genotyping data frame are given under 'Details'.

Usage

```
toArray(finalRep)
```

Arguments

`finalRep`	A data frame with the same structure as 'Final Report', containing genotyping data. Specifications are given under 'Details'.

Details

Used to turn Illumina's Genome Studio 'Final Report' file into an array. Specifications of the input data fram, 'finalRep', are:

- Input is a data frame with its first 4 columns listed in the following order:
  1. SNP names, as factor, with equal number of SNPs per individual.
  2. Identification codes, as factor, for genotyped individuals.

3. Allele 1 (one character: A, C, G, T, or -).
4. Allele 2 (one character: A, C, G, T, or -).

- All SNPs of individual 1 are listed first followed by SNPs of individual 2, and so on.
- SNPs are listed for each individual in the same order.

Note that it is easy to read the 'Final Report' file into a data frame which is then used as an input to `toArray`. See 'Examples'.

Value

An object of class matrix with 'number of rows = number of individuals' and 'number of columns = number of SNPs'. Each element of the matrix consists of a two-character string for the two DNA bases of a single SNP locus.

See Also

`arrayAppend`

Examples

```
## Read file './Final.Report', located in the current working directory,
## and place the first 4 columns in a data frame.

#d <- read.table("./Final.Report", skip=10)[,1:4]

## Use toArray to turn data read into a matrix

#ga <- toArray(finalRep = d)
#ga[1:6, 4000:4002]

#
#                SNP.4000           SNP.4001           SNP.4002
#Individual.1        "GG"               "CC"               "CC"
#Individual.2        "TG"               "AC"               "CC"
#Individual.3        "TG"               "AA"               "CC"
#Individual.4        "GG"               "AC"               "TC"
#Individual.5        "GG"               "AC"               "CC"
#Individual.6        "GG"               "AA"               "CC"
#
```

Index