

Package ‘GENLIB’

September 16, 2021

Type Package

Title Genealogical Data Analysis

Version 1.1.5

Date 2021-09-08

Description Genealogical data analysis including descriptive statistics (e.g., kinship and inbreeding coefficients) and gene-dropping simulations. See: “GENLIB: an R package for the analysis of genealogical data” Gauvin et al. (2015) <[doi:10.1186/s12859-015-0581-5](https://doi.org/10.1186/s12859-015-0581-5)>.

License GPL (>= 2)

LazyLoad yes

Depends R (>= 3.1.0), Rcpp (>= 0.9.10)

Imports kinship2, methods, bootstrap, Matrix, lattice, quadprog, foreach, parallel, doParallel

LinkingTo Rcpp

NeedsCompilation yes

Author Louis Houde [aut],
Jean-Francois Lefebvre [aut],
Valery Roy-Lagace [aut],
Sebastien Lemieux [aut],
Michael J. Fromberger [ctb],
Jarno van der Kolk [ctb],
Mohan Rakesh [ctb],
Marie-Helene Roy-Gagnon [cre]

Maintainer Marie-Helene Roy-Gagnon <mroygagn@uottawa.ca>

Repository CRAN

Date/Publication 2021-09-16 07:30:18 UTC

R topics documented:

GenlibR-package	3
Classes of basic handling of genealogy data	3
Classes of GLgroup handling	4

gen.branching	6
gen.children	7
gen.climbPAR	8
gen.completeness	8
gen.completenessVar	9
gen.depth	11
gen.f	11
gen.fCI	13
gen.find.Min.Distance.MRCA	14
gen.findDistance	15
gen.findFounders	16
gen.findMRCA	17
gen.founder	18
gen.gc	19
gen.genealogy	20
gen.genout	21
gen.getAncestorsPAR	22
gen.getFoundersPAR	23
gen.graph	23
gen.half.founder	25
gen.implex	26
gen.implexVar	27
gen.lineages	28
gen.max	29
gen.mean	30
gen.meangendepth	31
gen.meangendepthVar	32
gen.min	33
gen.nochildren	34
gen.noind	35
gen.nomen	36
gen.nowomen	36
gen.occ	37
gen.parent	38
gen.phi	39
gen.phiCI	40
gen.phiMean	41
gen.phiOver	42
gen.pro	43
gen.rec	44
gen.sibship	45
gen.simuHaplo	46
gen.simuProb	48
gen.simuSample	49
gen.simuSampleFreq	50
gen.simuSet	51
genea140	53
geneaJi	54

Classes of basic handling of genealogy data 3

GLgen-class	55
GLgroup-class	55
GLmultiList-class	56
pop140	56

Index 58

GenlibR-package *Genealogical Data Analysis*

Description

Genealogical data analysis including descriptive statistics (e.g., kinship and inbreeding coefficients) and gene-dropping simulations.

Details

Package: GenlibR
Type: Package
Version: 1.1.4
Date: 2021-09-15
License: GPL (>=2)
LazyLoad: yes

Author(s)

Louis Houde, Jean-Francois Lefebvre, Valery Roy-Lagace, Sebastien Lemieux

Classes of basic handling of genealogy data
Classes "GLmultiVector" "GLmultiArray4" "GLmultiMatrix"
"GLmultiNumber"

Description

Objects created to carry information mostly between invisible functions.

Objects from the Class

Objects can be created by calls of the form `new("GLmultiVector", 'matrix', depth = 'integer')`
`new("GLmultiArray4", 'Array', depth = 'integer')`
`new("GLmultiMatrix", 'Array', depth = 'integer')`
`new("GLmultiNumber", 'numeric', depth = 'integer', .Names = 'character')`

Slots

```

GLmultiVector
  Object of class "matrix"
  .Data: Object of class "integer"
    GLmultiArray4, GLmultiMatrix
  .Data: Object of class "array"
  depth: Object of class "integer"
    GLmultiNumber
  .Data: Object of class "numeric" ~~
  depth: Object of class "integer" ~~
  .Names: Object of class "character" ~~

```

Extends

Class `"matrix"`, from data part. Class `"array"`, by class `"matrix"`, distance 2. Class `"structure"`, by class `"matrix"`, distance 3. Class `"vector"`, by class `"matrix"`, distance 4, with explicit coerce.

Methods

```

[<- signature(x = "GLmultiVector", i = "ANY", j = "ANY", value = "ANY"): ...
[ signature(x = "GLmultiVector", i = "ANY", j = "ANY", drop = "ANY"): ...
depth signature(x = "GLmultiVector"): ...
Dim signature(object = "GLmultiVector"): ...

```

Examples

```

showClass("GLmultiVector")
showClass("GLmultiArray4")
showClass("GLmultiMatrix")
showClass("GLmultiNumber")

```

Classes of GLgroup handling

<i>Classes</i>	"GLmultiPhiGroup"	"GLmultiPhiGroupSingle"
	"GLmultiFGroup"	"GLmultiFGroupSingle"
	"GLCGMatrixGroupSingle"	

Description

Objects used to carry information mostly between invisible functions.

Objects from the Class

Objects can be created by calls of the form

```
new("GLmultiPhiGroup",MatriceCG,group = 'GLgroup',grindex = 'list')
new("GLmultiPhiGroupSingle",MatriceCG,group = 'GLgroup',grindex = 'list')
new("GLmultiFGroup",MatriceCG,group = 'GLgroup',grindex = 'list')
new("GLmultiFGroupSingle",MatriceCG,group = 'GLgroup',grindex = 'list')
new("GLCGMatrixGroupSingle",MatriceCG,group = 'GLgroup',grindex = 'list')
```

Slots

GLCGMatrixGroupSingle, GLmultiFGroupSingle, GLmultiPhiGroupSingle

Object of class "matrix"

.Data: Object of class "GLgroup"

grindex: Object of class "list" containing the indices of the probands of 'group'

GLmultiPhiGroup :

.Data: Object of class "GLmultiMatrix"

group: Object of class "GLgroup"

grindex: Object of class "list" containing the indices of the probands of 'group'

GLmultiFGroup :

.Data: Object of class "GLmultiVector"

group: Object of class "GLgroup"

grindex: Object of class "list" containing the indices of the probands of 'group'

Extends

Class "matrix", from data part. Class "array", by class "matrix", distance 2. Class "structure", by class "matrix", distance 3. Class "vector", by class "matrix", distance 4, with explicit coerce.

Methods

```
[<- signature(x = "GLCGMatrixGroupSingle", i = "ANY", j = "ANY", value = "ANY"): ...
```

```
[ signature(x = "GLCGMatrixGroupSingle", i = "ANY", j = "ANY", drop = "ANY"): ...
```

```
group signature(x = "GLCGMatrixGroupSingle"): ...
```

Examples

```
showClass("GLCGMatrixGroupSingle")
showClass("GLmultiFGroup")
showClass("GLmultiFGroupSingle")
showClass("GLmultiPhiGroupSingle")
showClass("GLmultiPhiGroup")
```

gen.branching	<i>Genealogy subset</i>
---------------	-------------------------

Description

Creates an object of class GLgen from an existing GLgen object by selecting specified individuals.

Usage

```
gen.branching( gen, pro = 0, ancestors = gen.founder(gen), bflag = 0)
```

Arguments

gen	An object of class GLgen obtained with gen.genealogy, gen.lineages or gen.branching. Required.
pro	Vector of proband id numbers to be included. Default is 0, which will select all individuals without children.
ancestors	Vector of ancestors id numbers to be included. Default includes all founders in the original GLgen object.
bflag	If set to 0 (default and recommended), probands and founders are filtered to quicken the calculations.

Value

returns a GLgen object

Note

Note that if there are no links between some probands and ancestors, these ids will not be included.

See Also

[gen.genealogy](#) [gen.graph](#) [gen.lineages](#)

Examples

```
data(geneaJi)
genJi<-gen.genealogy(geneaJi)
genJi_part<-gen.branching(genJi,pro=c(2,28))

# Plots of original genealogy and of the branched version
layout(matrix(1:2,c(1,2),byrow=TRUE))
gen.graph(genJi)
mtext("Original", line=2, cex=1.2)
gen.graph(genJi_part)
mtext("Branched tree\nfor individuals 2 and 28", line=1, cex=1.2)
```

gen.children	<i>Get id numbers of children</i>
--------------	-----------------------------------

Description

Returns the id numbers of the children of specified individuals.

Usage

```
gen.children( gen, individuals, ...)
```

Arguments

gen	An object of class GLgen obtained with gen.genealogy, gen.lineages or gen.branching. Required.
individuals	Vector of individual id numbers. Required.
...	Option to pass additionnal arguments automaticaly between methods. Internal use only.

Value

returns a vector of integer

See Also

[gen.genealogy](#) [gen.pro](#) [gen.founder](#) [gen.sibship](#) [gen.parent](#)

Examples

```
data(geneaJi)
genJi<-gen.genealogy(geneaJi)
gen.children(genJi, individuals=14)

data(genea140)
gen140<-gen.genealogy(genea140)
children51052<-gen.children(gen140,51052)
gen.graph(gen140, pro=c(children51052))
```

gen.climbPAR *Internal function used by gen.findMRCA*

Description

gen.climbPAR is an internal function used by gen.findMRCA.

Usage

```
gen.climbPAR(gen, individuals, founder)
```

Arguments

gen	The genealogy to consider.
individuals	probands to consider
founder	the founder by which the distance is calculated

Value

returns a list containing the founder ID and the distance.

See Also

[gen.findMRCA](#) [gen.getAncestorsPAR](#)

Examples

```
data(geneaJi)
genJi<-gen.genealogy(geneaJi)
gen.findMRCA(genJi, individuals=c(1,29), NbProcess = 1)
```

gen.completeness *Completeness*

Description

Returns the completeness of the genealogical data for the specified probands

Usage

```
gen.completeness( gen, pro="0", genNo=-1, type="MEAN", ...)
```


Arguments

gen	An object of class GLgen obtained with gen.genealogy, gen.lineages or gen.branching. Required.
pro	Vector of proband id numbers to be included. Default is 0, which will select all individuals without children.
genNo	Vector of generation numbers at which completeness should be calculated. Default is -1, which calculates completeness at each generation.
type	If type="MEAN" (default), mean completeness over all specified probands is calculated. If type="IND", completeness is calculated for each specified proband.
...	Option to pass additional arguments automatically between methods. Internal use only.

Value

A data frame with each generation at which the completeness is calculated as rows and one column when type is "MEAN". When type is "IND", the number of columns equals the number of probands specified.

References

Cazes P, Cazes MH. (1996) Comment mesurer la profondeur genealogique d'une ascendance? Population (French Ed) 51:117-140.

See Also

[gen.genealogy](#) [gen.rec](#) [gen.occ](#) [gen.implex](#) [gen.meangendepth](#)

Examples

```
data(geneaJi)
genJi<-gen.genealogy(geneaJi)
gen.completeness(genJi, type="IND")
# For the 4th generation
gen.completeness(genJi, type="IND", genNo=4)

data(genea140)
gen140<-gen.genealogy(genea140)
gen.completeness(gen140)
```

gen.completenessVar *Variance of completeness index*

Description

Returns the variance of the completeness index across probands.

Usage

```
gen.completenessVar( gen, pro = "0", genNo = -1, ...)
```

Arguments

gen	An object of class GLgen obtained with <code>gen.genealogy</code> , <code>gen.lineages</code> or <code>gen.branching</code> . Required.
pro	Vector of proband id numbers to be included. Default is 0, which will select all individuals without children.
genNo	Vector of generation numbers at which completeness should be calculated. Default is -1, which calculates completeness at each generation.
...	Option to pass additional arguments automatically between methods. Internal use only.

Value

A data frame with one column and each generation at which the variance is calculated as rows.

References

Cazes P, Cazes MH. (1996) Comment mesurer la profondeur genealogique d'une ascendance? Population (French Ed) 51:117-140.

See Also

[gen.gc](#)

Examples

```
data(geneaJi)
genJi<-gen.genealogy(geneaJi)
gen.completeness(genJi, type="IND")
gen.completenessVar(genJi)
# For the 4th generation
gen.completenessVar(genJi, genNo=4)
```

```
data(genea140)
gen140<-gen.genealogy(genea140)
gen.completeness(gen140)
gen.completenessVar(gen140)
```

gen.depth	<i>Genealogy depth</i>
-----------	------------------------

Description

Returns the number of generations in the genealogy from a GLgen object.

Usage

```
gen.depth(gen)
```

Arguments

gen	An object of class GLgen obtained with gen.genealogy, gen.lineages or gen.branching. Required.
-----	--

Value

returns a vector of integer

See Also

[gen.genealogy](#) [gen.min](#) [gen.mean](#) [gen.min](#) [gen.meangendepth](#)

Examples

```
data(geneaJi)
genJi<-gen.genealogy(geneaJi)
gen.depth(genJi)
```

```
data(genea140)
gen140<-gen.genealogy(genea140)
gen.depth(gen140)
```

gen.f	<i>Inbreeding coefficient</i>
-------	-------------------------------

Description

Returns the inbreeding coefficients of the specified probands

Usage

```
gen.f( gen, pro, depthmin= (gen.depth(gen)-1), depthmax= (gen.depth(gen)-1))
```

Arguments

gen	An object of class GLgen obtained with gen.genealogy, gen.lineages or gen.branching. Required.
pro	Vector of proband id numbers to be included.
depthmin	Minimum genealogical depth to consider in the calculation. Default is the whole genealogy.
depthmax	Maximum genealogical depth to consider in the calculation. Default is the whole genealogy.

Value

A vector or GLmultiVector object depending on the number of generations treated. GLmultiVector is an array of vectors, one for each depth. Array of size ('depthMax'-'depthMin'), vectors of size 'length(pro)'. A vector of size 'length(pro)' is returned when ('depthMax'-'depthMin') equals 1.

References

- Malecot G. (1948) Les mathematiques de l'heredite. Paris: Masson, p 65.
- Thompson EA. (1986) Pedigree Analysis in Human Genetics. Baltimore, MD, USA: Johns Hopkins University Press, p 25.
- Karigl G. (1981) A recursive algorithm for the calculation of identity coefficients. Ann Hum Genet 45:299-305.

See Also

[gen.genealogy](#) [gen.phi](#)

Examples

```
data(geneaJi)
genJi<-gen.genealogy(geneaJi)
f_values<-gen.f(genJi)
f_values
f_allgen<-gen.f(genJi, depthmin=1)
f_allgen<-unclass(f_allgen)

plot(1:7,f_allgen[1,],type="b",xlab="Generation",ylab="Inbreeding values",ylim=c(0,0.25),pch=0)
points(1:7,f_allgen[3,], type="b", lty=12, pch=1)
legend("topright", legend=c("Individual 1", "Individual 29"),lty=c(1,12), pch=c(0,1))

data(genea140)
gen140<-gen.genealogy(genea140)
inbreeding_val<-gen.f(gen140)
boxplot(inbreeding_val, horizontal=TRUE, xlab="Inbreeding values")
```

gen.fCI	<i>Average inbreeding coefficient confidence interval</i>
---------	---

Description

Returns the confidence interval of the average inbreeding among specified individuals

Usage

```
gen.fCI(vectF, prob=c(0.025,0.05,0.95,0.975), b="5000", print.it=FALSE)
```

Arguments

vectF	A vector of inbreeding coefficients obtained for example with gen.f. Required.
prob	Confidence limits probabilities. Default is probs=c(0.025, 0.05, 0.95, 0.975).
b	Number of simulations used to calculate the confidence interval. Default is b="5000".
print.it	If print.it=FALSE (default), progression in the number of simulations performed is not displayed.

Value

A vector or a GLmultiVector object depending on the type of 'vectF' entered. If 'vectF' is a GLmultiVector, an array of vectors (GLmultiVector), one for each depth, is returned. Array of size ('depthMax'-'depthMin'), vectors of size 'length(probs)'. If 'vectF' is a GLmultiVector, a vector of size 'length(probs)' is returned.

See Also

[gen.f](#)

Examples

```
data(genea140)
gen140<-gen.genealogy(genea140)
gen10 <-gen.branching(gen140, pro=gen.pro(gen140)[c(1:10)])
fval<-gen.f(gen10)
gen.fCI(fval)
```

gen.find.Min.Distance.MRCA

Finds the minimum distances between pairs of individuals given a set of ancestors.

Description

Returns the shortest distances (number of meioses) between pairs of probands given the matrix of MRCAs output by the gen.findMRCA function.

Usage

```
gen.find.Min.Distance.MRCA(genMatrix, individuals="ALL", ancestors="ALL")
```

Arguments

genMatrix	Matrix of most recent common ancestors, MRCAs, obtained with gen.findMRCA. Required.
individuals	Vector of proband id numbers to include. All are included by default.
ancestors	Vector of MRCA id numbers to include. All are included by default.

Value

returns a matrix

See Also

[gen.genealogy](#) [gen.founder](#) [gen.findMRCA](#) [gen.findDistance](#) [gen.findFounders](#)

Examples

```
data(geneaJi)
genJi<-gen.genealogy(geneaJi)
matMRCA<-gen.findMRCA(genJi, individuals=c(1,29), NbProcess = 1)
gen.find.Min.Distance.MRCA(matMRCA)

# Increasing NbProcess will decrease execution time
data(genea140)
gen140<-gen.genealogy(genea140)
matMRCA<-gen.findMRCA(gen140, individuals=c(409033,408728,408828), NbProcess = 1)
gen.find.Min.Distance.MRCA(matMRCA, individuals=c(409033,408728))
```

gen.findDistance	<i>Minimum genetic distance between two individuals</i>
------------------	---

Description

Returns the minimum distance (number of meioses) between the specified individuals through one specified ancestor.

Usage

```
gen.findDistance(gen, individuals, ancestor)
```

Arguments

gen	An object of class GLgen obtained with gen.genealogy, gen.lineages or gen.branching. Required.
individuals	A vector of individual id numbers between which to calculate the distance. Required.
ancestor	A common ancestor to the specified individuals. Required.

Value

returns a numeric value

See Also

[gen.genealogy](#) [gen.founder](#) [gen.findMRCA](#) [gen.findFounders](#) [gen.find.Min.Distance.MRCA](#)

Examples

```
data(geneaJi)
genJi<-gen.genealogy(geneaJi)
gen.findDistance(genJi, individuals=c(1,29), ancestor=17)
gen.findDistance(genJi, individuals=c(1,29), ancestor=26)

data(genea140)
gen140<-gen.genealogy(genea140)
gen.findDistance(gen140, individuals=c(409033,408728), ancestor=38714)
gen.findDistance(gen140, individuals=c(408728,408828), ancestor=95080)
```

gen.findFounders *Find common founder ancestors*

Description

Returns all the ancestors that are founders for specified individuals in the genealogy.

Usage

```
gen.findFounders(gen, individuals, NbProcess=parallel::detectCores()-1)
```

Arguments

gen	An object of class GLgen obtained with gen.genealogy, gen.lineages or gen.branching. Required.
individuals	Vector of individual id numbers for which to find the founders. Required
NbProcess	Number of processes to use when running this function. Default=parallel::detectCores()-1

Value

A vector of integers

Note

Uses slave processes to make the search faster. Those slave processes launch the function gen.getFoundersMPI.

See Also

[gen.genealogy](#) [gen.founder](#) [gen.findMRCA](#) [gen.findDistance](#) [gen.find.Min.Distance.MRCA](#)

Examples

```
data(geneaJi)
genJi<-gen.genealogy(geneaJi)
gen.founder(genJi)
gen.findFounders(genJi, individuals=c(1,29), NbProcess = 1)
```

```
data(genea140)
gen140<-gen.genealogy(genea140)
gen2<-gen.branching(gen140,pro=c(409033,408728))
# 615 founders in genealogy of #409033 and #408728
all_founders<-gen.founder(gen2)
length(all_founders)
```

```
all_commonFounders<-gen.findFounders(gen140, individuals=c(409033,408728), NbProcess = 1)
length(all_commonFounders) # 127 founders common to #409033 and #408728
```

gen.findMRCA *Finding most recent common ancestors, MRCA*

Description

Returns MRCA of pairs of specified individuals and the distance (number of meioses) between individuals through the MRCA.

Usage

```
gen.findMRCA(gen, individuals, NbProcess=parallel::detectCores()-1)
```

Arguments

gen	An object of class GLgen obtained with gen.genealogy, gen.lineages or gen.branching. Required.
individuals	Vector of individual id numbers for which to find the MRCA. Required
NbProcess	Number of processes to use when running this function. Default=parallel::detectCores()-1

Value

returns a matrix

Note

This function uses the parallel programming functions of foreach, snow and doSNOW for launching several processes.

See Also

[gen.genealogy](#) [gen.founder](#) [gen.findFounders](#) [gen.findDistance](#) [gen.find.Min.Distance.MRCA](#)

Examples

```
data(geneaJi)
genJi<-gen.genealogy(geneaJi)
gen.findMRCA(genJi, individuals=c(1,29), NbProcess = 1)

# For a more complex example:
data(genea140)
gen140<-gen.genealogy(genea140)
all_commonFounders<-gen.findFounders(gen140, individuals=c(409033,408728), NbProcess = 1)
# 127 founders common to #409033 and #408728
length(all_commonFounders)
# 22 most recent common ancestors for #409033 and #408728
MRCA_2ind<-gen.findMRCA(gen140, individuals=c(409033,408728), NbProcess = 1)
```

gen.founder	<i>Get founder id numbers</i>
-------------	-------------------------------

Description

Returns the id numbers of the founders. Founders are defined as the individuals without parents in the genealogy (i.e., mother id=0, father id=0).

Usage

```
gen.founder( gen, ...)
```

Arguments

gen	An object of class GLgen obtained with <code>gen.genealogy</code> , <code>gen.lineages</code> or <code>gen.branching</code> . Required.
...	Option to pass additionnal arguments automaticaly between methods. Internal use only.

Value

A vector of integer

See Also

[gen.genealogy](#) [gen.pro](#) [gen.half.founder](#) [gen.parent](#)

Examples

```
data(geneaJi)
genJi<-gen.genealogy(geneaJi)
# There are 6 founders
gen.founder(genJi)

data(genea140)
gen140<-gen.genealogy(genea140)
founder140<-gen.founder(gen140)
# There are 7399 founders
length(founder140)
```

gen.gc *Genetic contribution of ancestors*

Description

Returns the genetic contribution of ancestors to the gene pool of sepcific probands

Usage

```
gen.gc(gen, pro=0, ancestors=0, vctProb=c(0.5,0.5,0.5,0.5), typeCG="IND")
```

Arguments

gen	An object of class GLgen obtained with gen.genealogy, gen.lineages or gen.branching. Required.
pro	Vector of proband id numbers to be included. Default is 0, which will select all individuals without children.
ancestors	Vector of ancestors id numbers to be included. Default is 0, which will select all individuals without parents.
vctProb	Vector of transmission probabilities. The first two values indicate the probabilities of transmission of a father to his son and daughter, respectively, and the following two values are the same for the mother. Default is vctProb=c(0.5,0.5,0.5,0.5).
typeCG	IF typeCG="IND" (default), the genetic contribution from each ancestor is calculated for each proband. If typeCG="MEAN", the average (over all probands) genetic contributions of each ancestor is returned. If typeCG="PRODUCT", the product (over all probands) of genetic contributions is returned for each ancestor. If typeCG="TOTAL", the sum (over all probands) of genetic contributions is returned for each ancestor. If typeCG="CUMUL", ancestors are ranked in decreasing order of total contribution and cumulative contribution is returned.

Value

A matrix with rows corresponding to probands and columns corresponding to ancestors when typeCG="IND". For the other typeCG values, rows are ancestors and there is one column containing mean, product, total or cumulative values.

References

- Roberts DF. (1968) Genetic effects of population size reduction. *Nature*, 220, 1084-1088.
- O'Brien E, Jorde LB, Ronnlof B, Fellman JO, Eriksson AW. (1988) Founder effect and genetic disease in Sottunga, Finland. *American Journal of Physical Anthropology*, 77, 335-346.

See Also

[gen.genealogy](#) [gen.rec](#) [gen.occ](#) [gen.implex](#) [gen.meangendepth](#) [gen.completeness](#)

Examples

```

data(geneaJi)
genJi<-gen.genealogy(geneaJi)
gc<-gen.gc(genJi, pro=c(1,29), ancestors=c(3,6,10,12,14,16,18,20,26,28))
gc
gc_cum<-gen.gc(genJi, pro=c(1,29), ancestors=c(3,6,10,12,14,16,18,20,26,28), type="CUMUL")
gc_cum

data(genea140)
gen140<-gen.genealogy(genea140)
gc<-gen.gc(gen140, pro=c(454422,676521,677273,717537,717634,717709,868572),
           ancestors=c(18305,18528,31114,18325))

gc
# Mother-daughter transmission only with probability=0.5
gc_MoLi<-gen.gc(gen140, pro=c(217891,302718,409282,802424,409682,443151),
               ancestors=c(18321,218231,296200,39066,18679,442607), vctProb=c(0,0,0,0.5))
gc_MoLi

```

gen.genealogy

Create object of class GLgen

Description

Creates an object of class GLgen that contains the ascending genealogies derived from input data in pedigree format

Usage

```
gen.genealogy(ped, autoComplete=FALSE, ...)
```

Arguments

ped	A matrix or data frame with the following columns: individual id, father id, mother id, sex. Required. Individual id numbers must be numeric and unique. If an individual does not have a father and/or mother, the father and/or mother id numbers must be set to 0. Sex must be equal to M or 1 for males and F or 2 for females. The sex column is optional for this function but necessary for some other functions using GLgen objects.
autoComplete	If TRUE, any non-zero father and mother id numbers not appearing in the individual id column, will be added in the individual column as having no father or mother (both set to 0). Default to FALSE.
...	Option to pass additional arguments automatically between methods. Internal use only.

Value

An object of class GLgen.

See Also

[gen.graph](#) [gen.genout](#)

Examples

```
ind<-c(1,2,3,11:21,101:108,201:202)
father<-c(11,15,15,102,0,102,0,103,103,103,105,105,107,107,0,202,0,202,202,0,202,0,0,0)
mother<-c(12,14,14,101,0,101,0,104,104,104,106,106,108,108,0,201,0,201,201,0,201,0,0,0)
sex<-c(1,2,2,1,2,1,2,1,2,1,2,2,2,1,2,1,1,2,1,2,1,2,2,1)
gen.df<-data.frame(ind, father, mother, sex)
gen.genealogy(gen.df)

data(geneaJi)
geneaJi[1:5,]
genJi<-gen.genealogy(geneaJi)
# Print basic information about the genealogy
genJi
```

gen.genout

Create pedigree data

Description

Creates data frame in pedigree format from an object of class GLgen

Usage

```
gen.genout( gen, sorted = FALSE)
```

Arguments

gen	An object of class GLgen obtained with <code>gen.genealogy</code> , <code>gen.lineages</code> or <code>gen.branching</code> . Required.
sorted	If <code>sorted=FALSE</code> (default), individual id number are not sorted in output data. Id numbers for parents will be placed before their children ids. If <code>sorted=TRUE</code> , individual id numbers will be sorted.

Value

returns a data.frame containing the following: **ind father mother sex**

See Also

[gen.genealogy](#) [gen.graph](#) [gen.branching](#) [gen.lineages](#)

Examples

```
data(geneaJi)
# Original data is a data.frame
geneaJi[1:12,]

genJi<-gen.genealogy(geneaJi)
# As a genealogy object
genJi

# Genealogy as a data.frame
genJi_df<-gen.genout(genJi)
genJi_df[1:12,]

# Maternal lineage
genJi_MaLi<-gen.lineages(geneaJi, maternal = TRUE)
# Maternal lineage as a data.frame
genJi_MaLi_df<-gen.genout(genJi_MaLi)
genJi_MaLi_df
```

gen.getAncestorsPAR *Internal function used by gen.findMRCA*

Description

gen.getAncestorsPAR is an internal function used by *gen.findMRCA*.

Usage

```
gen.getAncestorsPAR(gen, pro)
```

Arguments

<code>gen</code>	The genealogy to consider.
<code>pro</code>	probands to consider

Value

Vector of the founders identity.

See Also

[gen.findMRCA](#) [gen.climbPAR](#)

Examples

```
data(geneaJi)
genJi<-gen.genealogy(geneaJi)
gen.findMRCA(genJi, individuals=c(1,29), NbProcess = 1)
```

gen.getFoundersPAR	<i>Internal function used by gen.findFounders</i>
--------------------	---

Description

gen.getFoundersPAR is an internal function used by gen.findFounders.

Usage

```
gen.getFoundersPAR(gen, pro)
```

Arguments

gen	The genealogy to consider.
pro	probands to consider

Value

A vector of the founders identity.

See Also

[gen.findFounders](#) [gen.findDistance](#)

Examples

```
data(geneaJi)
genJi<-gen.genealogy(geneaJi)
gen.founder(genJi, individuals=c(1,29))
gen.findFounders(genJi, individuals=c(1,29), NbProcess = 1)
```

gen.graph	<i>Pedigree graphical tool</i>
-----------	--------------------------------

Description

Function that plots pedigrees of GLgen objects.

Usage

```
gen.graph( gen, pro=gen.pro(gen), ancestors=gen.founder(gen),
  indVarAffected=gen.genout(gen)$ind, varAffected=gen.genout(gen)$ind, cex="1",
  col="0", symbolsize="1", width="1", packed=FALSE, align=TRUE, ...)
```

Arguments

gen	An object of class GLgen obtained with gen.genealogy, gen.lineages or gen.branching. Required.
pro	Vector of proband id numbers to be included. Default is 'gen.pro(gen)', which will select all individuals without children.
ancestors	Vector of ancestors id numbers to be included. Default is 'gen.founder(gen)', which will select all individuals without parents.
indVarAffected	Vector of individuals id numbers used for labeling. Default is 'gen.genout(gen)\$ind', which is all the individuals of the genealogy.
varAffected	Vector of individuals labels. Default is 'gen.genout(gen)\$ind', meaning that the individuals labels are their ids.
cex	Controls text size (same as kinship2::plot.pedigree). Default is 1.
col	Color for each id (same as kinship2::plot.pedigree). Default is 0, which assigns the same color to everyone.
symbolsize	Controls symbolsize (same as kinship2::plot.pedigree). Default is 1.
width	For a packed pedigree, the minimum width allowed in the realignment of pedigrees (same as kinship2::plot.pedigree). Default is 1.
packed	If TRUE, uniform distance between all individuals at a given level (same as kinship2::plot.pedigree). Default is FALSE.
align	Controls the extra effort spent trying to align children underneath parents, but without making the pedigree too wide (same as kinship2::plot.pedigree). Default is TRUE.
...	Other arguments that can be passed to the kinship2::plot.pedigree.

Value

Returns the same invisible list as that returned by kinship2::plot.pedigree, which contains the following: **plist** list containing the information about the pedigree (n, id, pos, fam, spouse) **x** x-axis position **y** y-axis position **boxw** box width **boxh** box height **call** the call made to plot.pedigree() (kinship2 package)

See Also

[gen.genealogy](#) [gen.genout](#) [gen.branching](#)

Examples

```
ind<-c(1,2,3,11:21,101:108,201:202)
father<-c(11,15,15,102,0,102,0,103,103,0,105,105,107,107,0,202,0,202,202,0,202,0,0,0)
mother<-c(12,14,14,101,0,101,0,104,104,0,106,106,108,108,0,201,0,201,201,0,201,0,0,0)
sex<-c(1,2,2,1,2,1,2,1,2,1,2,2,2,1,2,1,1,2,1,2,1,2,2,1)
gen.df<-data.frame(ind, father, mother, sex)
genEx<-gen.genealogy(gen.df)
gen.graph(genEx)

data(geneaJi)
```



```
geneaJi[1:5,]  
genJi<-gen.genealogy(geneaJi)  
gen.graph(genJi)
```

gen.half.founder *Get half-founder id numbers*

Description

Returns the id numbers of the half-founders. Half-founders are defined as the individuals with only one known parent in the genealogy (i.e., either mother id=0 or father id=0).

Usage

```
gen.half.founder( gen, ...)
```

Arguments

<code>gen</code>	An object of class GLgen obtained with <code>gen.genealogy</code> , <code>gen.lineages</code> or <code>gen.branching</code> . Required.
<code>...</code>	Option to pass additional arguments automatically between methods. Internal use only.

Value

returns a vector of integer

See Also

[gen.genealogy](#) [gen.pro](#) [gen.founder](#) [gen.parent](#)

Examples

```
data(geneaJi)  
genJi<-gen.genealogy(geneaJi)  
# There are 2 half-founders  
gen.half.founder(genJi)
```

 gen.implex

Genealogical implex

Description

Returns the genealogical implex index (a measure of pedigree collapsing) for the specified probands.

Usage

```
gen.implex( gen, pro = "0", genNo = -1, type = "MEAN", onlyNewAnc = FALSE, ...)
```

Arguments

gen	An object of class GLgen obtained with gen.genealogy, gen.lineages or gen.branching. Required.
pro	Vector of proband id numbers to be included. Default is 0, which will select all individuals without children.
genNo	Vector of generation numbers at which the implex should be calculated. Default is -1, which calculates the implex at each generation.
type	If type="MEAN" (default), implex index values are averaged over all specified probands. If type="IND", the implex index is calculated for all specified proband together. If type="ALL", the implex index is calculated for each specified proband.
onlyNewAnc	If onlyNewAnc=FALSE (default), all ancestors will be considered. If onlyNewAnc=TRUE, only new ancestors will be counted (i.e., an ancestor is not counted again if it has already been counted in another generation).
...	Option to pass additionnal arguments automatically between methods. Internal use only.

Value

A data frame with each generation at which the completeness is calculated as rows and one column when type is "MEAN". When type is "IND", the number of columns equals the number of probands specified.

References

Cazes P, Cazes MH. (1996) Comment mesurer la profondeur genealogique d'une ascendance? Population (French Ed) 51:117-140.

See Also

[gen.genealogy](#) [gen.occ](#) [gen.rec](#) [gen.meangendepth](#) [gen.completeness](#) [gen.gc](#)

Examples

```

data(geneaJi)
genJi<-gen.genealogy(geneaJi)
gen.implex(genJi)
gen.implex(genJi, type="IND")
# For the 5th generation
gen.implex(genJi, type="IND", genNo=5)

data(genea140)
gen140<-gen.genealogy(genea140)
gen.implex(gen140)
gen.implex(gen140, pro=c(454422, 676521, 677273, 717537, 717634, 717709, 868572))
gen.implex(gen140, pro=c(454422, 676521, 677273, 717537, 717634, 717709, 868572), type="IND")

```

gen.implexVar	<i>Variance of genealogical implex</i>
---------------	--

Description

Returns the variance of the implex index across probands.

Usage

```
gen.implexVar( gen, pro = "0", onlyNewAnc = FALSE, genNo = -1, ...)
```

Arguments

gen	An object of class GLgen obtained with gen.genealogy, gen.lineages or gen.branching. Required.
pro	Vector of proband id numbers to be included. Default is 0, which will select all individuals without children.
onlyNewAnc	If onlyNewAnc=FALSE (default), all ancestors will be considered. If onlyNewAnc=TRUE, only new ancestors will be counted (i.e., an ancestor is not counted again if it has already been counted in another generation).
genNo	Vector of generation numbers at which the implex should be calculated. Default is -1, which calculates the implex at each generation.
...	Additional arguments to be passed to methods.

Value

A data frame with one column and each generation at which the variance is calculated as rows

References

Cazes P, Cazes MH. (1996) Comment mesurer la profondeur genealogique d'une ascendance? Population (French Ed) 51:117-140.

See Also[gen.gc](#)**Examples**

```
data(geneaJi)
genJi<-gen.genealogy(geneaJi)
gen.implex(genJi, type="IND")
gen.implexVar(genJi)
# For the 5th generation
gen.implexVar(genJi, genNo=5)
```

```
data(genea140)
gen140<-gen.genealogy(genea140)
gen.implex(gen140)
gen.implex(gen140, pro=c(454422, 676521, 677273, 717537, 717634, 717709, 868572), type="IND")
gen.implexVar(gen140, pro=c(454422, 676521, 677273, 717537, 717634, 717709, 868572), type="IND")
```

`gen.lineages`*Create object of class GLgen for maternal or paternal lineages*

Description

Creates an object of class GLgen that contains maternal or paternal lineages selected from input data in pedigree format

Usage

```
gen.lineages(ped, pro = "0", maternal = TRUE, ...)
```

Arguments

ped	A matrix or data frame with the following columns: individual id, father id, mother id, sex. Required. Individual id numbers must be numeric and unique. If an individual does not have a father and/or mother, the father and/or mother id numbers must be set to 0. All non-zero father and mother id numbers must also appear in the individual id column. Sex must be equal to M or 1 for males and F or 2 for females. The sex column is optional for this function but necessary for some other functions using GLgen objects.
pro	Vector of individual id numbers for which lineages should be included. Optional.
maternal	If mat=TRUE (default), maternal lineages are selected. mat=FALSE returns paternal lineages.
...	Option to pass additional arguments automatically between methods. Internal use only.

Value

returns a GLgen object

See Also

[gen.genealogy](#) [gen.graph](#) [gen.branching](#) [gen.genout](#)

Examples

```
data(geneaJi)
genJi <- gen.genealogy(geneaJi)
genJi_MaLi<-gen.lineages(geneaJi, maternal = TRUE)
genJi_FaLi<-gen.lineages(geneaJi, maternal = FALSE)

# Plots of original genealogy and maternal and paternal lineages
layout(matrix(1:3,c(1,3),byrow=TRUE), widths =c(3,1,1), heights = 1)
gen.graph(genJi)
mtext("Original", line=2)
gen.graph(genJi_MaLi)
mtext("Maternal\nlineages", line=1)
gen.graph(genJi_FaLi)
mtext("Paternal\nlineages", line=1)
```

gen.max

Maximum number of generations

Description

Returns the maximum number of generations between all probands and the individuals specified. Probands are defined as the individuals without children in the genealogy.

Usage

```
gen.max( gen, individuals)
```

Arguments

gen An object of class GLgen obtained with `gen.genealogy`, `gen.lineages` or `gen.branching`. Required.

individuals A vector of ids specifying the individuals to include in the calculation. Required.

Value

returns a vector of integer

See Also

[gen.genealogy](#) [gen.mean](#) [gen.min](#) [gen.depth](#) [gen.meangendepth](#)

Examples

```
data(geneaJi)
genJi<-gen.genealogy(geneaJi)
gen.min(genJi,c(17,26))
gen.mean(genJi,c(17,26))
gen.max(genJi,c(17,26))
```

```
data(genea140)
gen140<-gen.genealogy(genea140)
gen.min(gen140,c(18311,18430))
gen.mean(gen140,c(18311,18430))
gen.max(gen140,c(18311,18430))
```

gen.mean	<i>Mean number of generations</i>
----------	-----------------------------------

Description

Returns the average number of generations between all probands and the individuals specified. Probands are defined as the individuals without children in the genealogy.

Usage

```
gen.mean( gen, individuals)
```

Arguments

gen	An object of class GLgen obtained with gen.genealogy, gen.lineages or gen.branching. Required.
individuals	A vector of ids specifying the individuals to include in the calculation. Required.

Value

returns a numeric value

See Also

[gen.genealogy](#) [gen.min](#) [gen.max](#) [gen.depth](#) [gen.meangendepth](#)

Examples

```
data(geneaJi)
genJi<-gen.genealogy(geneaJi)
gen.min(genJi,c(17,26))
gen.mean(genJi,c(17,26))
gen.max(genJi,c(17,26))
```

```

data(genea140)
gen140<-gen.genealogy(genea140)
gen.min(gen140,c(18311,18430))
gen.mean(gen140,c(18311,18430))
gen.max(gen140,c(18311,18430))

```

gen.meangendepth	<i>Expected Genealogical Depth</i>
------------------	------------------------------------

Description

Returns the expected genealogical depth.

Usage

```
gen.meangendepth( gen, pro = "0", type = "MEAN", ...)
```

Arguments

gen	An object of class GLgen obtained with gen.genealogy, gen.lineages or gen.branching. Required.
pro	Vector of proband id numbers to be included. Default is 0, which will select all individuals without children.
type	If type="MEAN" (default), mean genealogical depth over all specified probands is calculated. If type="IND", mean genealogical depth is calculated for each specified proband.
...	Option to pass additionnal arguments automaticaly between methods. Internal use only.

Value

A data frame with only one numeric value when type is "MEAN". When type is "IND", the number of rows equals the number of probands specified.

References

Cazes P, Cazes MH. (1996) Comment mesurer la profondeur genealogique d'une ascendance? Population (French Ed) 51:117-140.

Kouladjian K. (1986) Une mesure d'entropie genealogique. Chicoutimi, SOREP, Document III-C-43.

De Brakaeleer M, Bellis G. (1994) Genealogies et reconstitutions de familles en genetique humaine. Dossiers et Recherches, no 43, INED, Paris.

See Also

[gen.genealogy](#) [gen.occ](#) [gen.implex](#) [gen.rec](#) [gen.completeness](#) [gen.gc](#)

Examples

```
data(geneaJi)
genJi<-gen.genealogy(geneaJi)
gen.meangendepth(genJi)
gen.meangendepth(genJi, type="IND")
```

```
data(genea140)
gen140<-gen.genealogy(genea140)
gen.meangendepth(gen140)
gen.meangendepth(gen140, pro=c(454422, 676521, 677273, 717537, 717634, 717709, 868572))
gen.meangendepth(gen140, pro=c(454422, 676521, 677273, 717537, 717634, 717709, 868572), type="IND")
```

`gen.meangendepthVar` *Variance of genealogical depth*

Description

Returns the variance of the genealogical depth

Usage

```
gen.meangendepthVar( gen, pro = "0", type = "MEAN", ...)
```

Arguments

<code>gen</code>	An object of class <code>GLgen</code> obtained with <code>gen.genealogy</code> , <code>gen.lineages</code> or <code>gen.branching</code> . Required.
<code>pro</code>	Vector of proband id numbers to be included. Default is 0, which will select all individuals without children.
<code>type</code>	If <code>type="MEAN"</code> (default), the average of genealogical depth variances (over all probands) is returned. If <code>type="IND"</code> , the variance of the genealogical depth is calculated for each specified proband.
<code>...</code>	Option to pass additional arguments automatically between methods. Internal use only.

Value

A data frame with only one numeric value when `type` is "MEAN". When `type` is "IND", the number of rows equals the number of probands specified.

References

Cazes P, Cazes MH. (1996) Comment mesurer la profondeur genealogique d'une ascendance? Population (French Ed) 51:117-140.

Kouladjian K. (1986) Une mesure d'entropie genealogique. Chicoutimi, SOREP, Document III-C-43.

De Brakaeleer M, Bellis G. (1994) Genealogies et reconstitutions de familles en genetique humaine. Dossiers et Recherches, no 43, INED, Paris.

See Also

[gen.gc](#)

Examples

```
data(geneaJi)
genJi<-gen.genealogy(geneaJi)
gen.meangendepth(genJi, type="IND")
gen.meangendepthVar(genJi, type="IND")
gen.meangendepthVar(genJi, type="MEAN")

data(genea140)
gen140<-gen.genealogy(genea140)
gen.meangendepth(gen140)
probands <- c(454422,676521,677273,717537,717634,717709,868572)
gen.meangendepth(gen140, pro=probands)
gen.meangendepthVar(gen140, pro=probands)
gen.meangendepthVar(gen140, pro=probands, type="MEAN")

gen.meangendepth(gen140, pro=probands, type="IND")
gen.meangendepthVar(gen140, pro=probands, type="IND")
```

gen.min

Minimum number of generations

Description

Returns the minimum number of generations between all probands and the individuals specified. Probands are defined as the individuals without children in the genealogy.

Usage

```
gen.min( gen, individuals)
```

Arguments

- `gen` An object of class `GLgen` obtained with `gen.genealogy`, `gen.lineages` or `gen.branching`. Required.
- `individuals` A vector of ids specifying the individuals to include in the calculation. Required.

Value

returns a vector of integer

See Also

[gen.genealogy](#) [gen.mean](#) [gen.max](#) [gen.depth](#) [gen.meangendepth](#)

Examples

```
data(geneaJi)
genJi<-gen.genealogy(geneaJi)
gen.min(genJi,c(17,26))
gen.mean(genJi,c(17,26))
gen.max(genJi,c(17,26))
```

```
data(genea140)
gen140<-gen.genealogy(genea140)
gen.min(gen140,c(18311,18430))
gen.mean(gen140,c(18311,18430))
gen.max(gen140,c(18311,18430))
```

<code>gen.nochildren</code>	<i>Number of children</i>
-----------------------------	---------------------------

Description

Returns the number of children for specified individuals

Usage

```
gen.nochildren( gen, individuals)
```

Arguments

- `gen` An object of class `GLgen` obtained with `gen.genealogy`, `gen.lineages` or `gen.branching`. Required.
- `individuals` A vector of ids of the individuals on each of whom the number of children is to be returned. Required.

Value

A vector containing the number of children for each individual specified.

See Also

[gen.genealogy](#) [gen.nowomen](#) [gen.noind](#) [gen.nomen](#) [gen.children](#)

Examples

```
data(geneaJi)
genJi<-gen.genealogy(geneaJi)
gen.children(genJi,14)
gen.nochildren(genJi,individuals=c(1,12,14,20))
```

```
data(genea140)
gen140<-gen.genealogy(genea140)
children51052<-gen.children(gen140,51052)
gen.nochildren(gen140,51052)
gen.graph(gen140, pro=children51052)
```

gen.noind	<i>Number of individuals</i>
-----------	------------------------------

Description

Returns the number of individuals included in the genealogy

Usage

```
gen.noind( gen)
```

Arguments

gen An object of class GLgen obtained with [gen.genealogy](#), [gen.lineages](#) or [gen.branching](#).
Required.

Value

returns a vector of integer

See Also

[gen.genealogy](#) [gen.nowomen](#) [gen.nochildren](#) [gen.nomen](#)

Examples

```
data(geneaJi)
genJi<-gen.genealogy(geneaJi)
gen.noind(genJi)
```

 gen.nomen

Number of men

Description

Returns the number of men included in the genealogy

Usage

```
gen.nomen( gen)
```

Arguments

gen An object of class GLgen obtained with gen.genealogy, gen.lineages or gen.branching. Required.

Value

returns a vector of integer

See Also

[gen.genealogy](#) [gen.nowomen](#) [gen.nochildren](#) [gen.noind](#)

Examples

```
data(geneaJi)
genJi<-gen.genealogy(geneaJi)
gen.nomen(genJi)
```

 gen.nowomen

Number of women

Description

Returns the number of women included in the genealogy

Usage

```
gen.nowomen( gen)
```

Arguments

gen An object of class GLgen obtained with gen.genealogy, gen.lineages or gen.branching. Required.

Value

returns a vector of integer

See Also

[gen.genealogy](#) [gen.nomen](#) [gen.nochildren](#) [gen.noind](#)

Examples

```
data(geneaJi)
genJi<-gen.genealogy(geneaJi)
gen.nowomen(genJi)
```

gen.occ

Ancestor occurrences

Description

Returns the number of times that the specified ancestors are present in the genealogies of the specified probands.

Usage

```
gen.occ( gen, pro = "0", ancestors = "0", typeOcc = "IND", ...)
```

Arguments

gen	An object of class GLgen obtained with gen.genealogy, gen.lineages or gen.branching. Required.
pro	Vector of proband id numbers to be included. Default is 0, which will select all individuals without children.
ancestors	Vector of ancestors id numbers to be included. Default is 0, which will select all individuals without parents.
typeOcc	If typeOcc="IND" (default), the number of occurrences for each proband will be returned. If typeOcc="TOTAL", the sum of all occurrences over all probands will be returned.
...	Option to pass additional arguments automatically between methods. Internal use only.

Value

A matrix with number of lines equal to the number of ancestors included and the number of columns equal to the number of probands included if typeOcc="BRUT" or only one column if typeOcc="TOTAL"

See Also

[gen.genealogy](#) [gen.rec](#) [gen.implex](#) [gen.meangendepth](#) [gen.gc](#)

Examples

```
data(geneaJi)
genJi<-gen.genealogy(geneaJi)
# Number of occurrences of ancestors in the specified proband's genealogy
gen.occ(genJi, pro=c(1,29), ancestors=c(3,6,10,12,14,16,18,20,26,28))

data(genea140)
gen140<-gen.genealogy(genea140)
gen.occ(gen140, pro=c(454422,677273,717537,717634,717709,868572),
        ancestors=c(18305,18528,31114,18325))
```

gen.parent

Get id numbers of parents

Description

Returns the id numbers of the parents of specified individuals.

Usage

```
gen.parent( gen, individuals, output = "FaMo", ...)
```

Arguments

gen	An object of class GLgen obtained with <code>gen.genealogy</code> , <code>gen.lineages</code> or <code>gen.branching</code> . Required.
individuals	Vector of individual id numbers. Required.
output	If <code>output="FaMo"</code> (default) then both mothers and fathers are included. "Mo" outputs mothers only and "Fa", fathers only.
...	Option to pass additional arguments automatically between methods. Internal use only.

Value

returns a list containing the following: **Fathers Mothers**

See Also

[gen.genealogy](#) [gen.pro](#) [gen.founder](#) [gen.children](#) [gen.sibship](#)

Examples

```
data(geneaJi)
genJi<-gen.genealogy(geneaJi)
gen.parent(genJi, individuals=c(3,21,29))
```

gen.phi	<i>Kinship coefficient</i>
---------	----------------------------

Description

Returns the kinship coefficients between pairs of individuals

Usage

```
gen.phi(gen, pro, depthmin=(gen.depth(gen)-1), depthmax=(gen.depth(gen)-1), MT=FALSE)
```

Arguments

gen	An object of class GLgen obtained with gen.genealogy, gen.lineages or gen.branching. Required.
pro	Vector of proband id numbers to be included. Required.
depthmin	Minimum genealogical depth to consider in the calculation. Default is the whole genealogy.
depthmax	Maximum genealogical depth to consider in the calculation. Default is the whole genealogy.
MT	Allows parallel computing when set to TRUE. Default is MT=FALSE.

Value

A matrix or a GLmultiMatrix object depending on the number of generations treated. GLmultiMatrix is an array of matrices, one for each depth. Array of size ('depthMax'-'depthMin') and matrices of size 'length(pro)' * 'length(pro)'. The matrix object, also of size 'length(pro)' * 'length(pro)', is returned when ('depthMax'-'depthMin') equals 1.

References

Malecot G. (1948) Les mathematiques de l'heredite. Paris: Masson, p 65.

Thompson EA. (1986) Pedigree Analysis in Human Genetics. Baltimore, MD, USA: Johns Hopkins University Press, p 25.

Karigl G. (1981) A recursive algorithm for the calculation of identity coefficients. Ann Hum Genet 45:299-305.

See Also

[gen.genealogy](#) [gen.f](#)

Examples

```

data(geneaJi)
genJi<-gen.genealogy(geneaJi)
kinship<-gen.phi(genJi)
kinship

kinship_allgenerations<-gen.phi(genJi, depthmin =1)
kinship_allgenerations
# 7th generations back in time is equivalent to considering all generations
kinship_allgenerations <- unclass(kinship_allgenerations)
kinship_allgenerations[,7]==kinship

kinship_allgenerations[1,2,]

# Plot of kinship varying according to number of generations considered
plot(1:7,kinship_allgenerations[1,2,], type="b", xlab="Generation", ylab="Kinship value",
     ylim=c(0,0.6), pch=0)
points(1:7,kinship_allgenerations[1,3,], type="b", lty=12, pch=1)
legend("topright", legend=c("Individuals 1 and 2", "Individuals 2 and 29"),lty=c(1,12), pch=c(0,1))

```

gen.phiCI

Average kinship confidence interval

Description

Returns the confidence interval of the average kinship among pairs of specified individuals

Usage

```
gen.phiCI(phiMatrix, prob=c(0.025,0.05,0.95,0.975), b=5000, print.it=FALSE)
```

Arguments

phiMatrix	A square matrix of kinship coefficients obtained for example with gen.phi, or an array of square matrices (GLmultiMatrix object) of kinship coefficients obtained with gen.phi. Required.
prob	Confidence limits probabilities. Default is probs=c(0.025, 0.05, 0.95, 0.975).
b	Number of simulations used to calculate the confidence interval. Default is b="5000".
print.it	If print.it=FALSE (default), progression in the number of simulations performed is not displayed.

Value

A vector or a GLmultiVector object depending on the type of 'phiMatrix' entered. If 'phiMatrix' is a GLmultiMatrix, an array of vectors (GLmultiVector), one for each depth, is returned. Array of size ('depthMax'-'depthMin'), vectors of size 'length(probs)'. If 'phiMatrix' is a matrix, a vector of size 'length(probs)' is returned.

See Also

[gen.genealogy](#) [gen.phi](#) [gen.phiOver](#) [gen.phiMean](#)

Examples

```
data(genea140)
gen140<-gen.genealogy(genea140)
ge07<-gen.branching(gen140, pro=gen.pro(gen140)[c(1:7)])
phiMat<-gen.phi(ge07)
gen.phiCI(phiMat)
```

gen.phiMean	<i>Average kinship</i>
-------------	------------------------

Description

Returns the average kinship among pairs of specified individuals

Usage

```
gen.phiMean( phiMatrix)
```

Arguments

phiMatrix A square matrix of kinship coefficients obtained for example with gen.phi. Required.

Value

returns a numeric value

See Also

[gen.genealogy](#) [gen.phi](#) [gen.phiOver](#) [gen.phiCI](#)

Examples

```
data(geneaJi)
genJi<-gen.genealogy(geneaJi)
kinship<-gen.phi(genJi)
gen.phiMean(kinship)

data(genea140)
gen140<-gen.genealogy(genea140)
phi6subjects<-gen.phi(gen140, pro=c(454422, 676521, 677273, 717537, 717634, 717709, 868572))
gen.phiMean(phi6subjects)
```

gen.phiOver	<i>Kinship above threshold</i>
-------------	--------------------------------

Description

Returns the pairs of individuals with kinship coefficient values greater than specified threshold.

Usage

```
gen.phiOver( phiMatrix, threshold)
```

Arguments

phiMatrix	A square matrix of kinship coefficients obtained for example with gen.phi. Required.
threshold	Threshold of kinship values to return.

Value

A data frame containing the probands and their kinship.

See Also

[gen.genealogy](#) [gen.phi](#) [gen.phiMean](#) [gen.phiCI](#)

Examples

```
data(geneaJi)
genJi<-gen.genealogy(geneaJi)
kinship<-gen.phi(genJi)
gen.phiOver(kinship, 0.1)

data(genea140)
gen140<-gen.genealogy(genea140)
phi9subj<-gen.phi(gen140,pro=c(408758,408950,409082,409111,676521,717537,717634,717709,868572))
gen.phiOver(phi9subj,0.025)
```

gen.pro	<i>Get proband id numbers</i>
---------	-------------------------------

Description

Returns the id numbers of the probands. Probands are defined as the individuals without children in the genealogy.

Usage

```
gen.pro( gen, ...)
```

Arguments

gen	An object of class GLgen obtained with <code>gen.genealogy</code> , <code>gen.lineages</code> or <code>gen.branching</code> . Required.
...	Option to pass additionnal arguments automaticaly between methods. Internal use only.

Value

returns a vector of integer

See Also

[gen.genealogy](#) [gen.founder](#) [gen.half.founder](#) [gen.parent](#) [gen.children](#)

Examples

```
data(geneaJi)
genJi<-gen.genealogy(geneaJi)
# There are 3 probands
gen.pro(genJi)
```

```
data(genea140)
gen140<-gen.genealogy(genea140)
gen.pro(gen140)
# There are 140 probands
```

gen.rec *Ancestors coverage*

Description

Returns the number of specified probands genealogically related to specified ancestors (i.e., ancestor occurs in the proband's genealogy).

Usage

```
gen.rec( gen, pro = "0", ancestors = "0", ...)
```

Arguments

gen	An object of class GLgen obtained with gen.genealogy, gen.lineages or gen.branching. Required.
pro	Vector of proband id numbers to be included. Default is 0, which will select all individuals without children.
ancestors	Vector of ancestors id numbers to be included. Default is 0, which will select all individuals without parents.
...	Option to pass additional arguments automatically between methods. Internal use only.

Value

A matrix with number of lines equal to the number of ancestors specified and one column.

Note

If an ancestor is also a proband, he/she will be counted in his/her coverage.

See Also

[gen.genealogy](#) [gen.occ](#) [gen.implex](#) [gen.meangendepth](#) [gen.gc](#)

Examples

```
data(geneaJi)
genJi<-gen.genealogy(geneaJi)
# Number of probands which are descendants of an ancestor
gen.rec(genJi)
```

```
data(genea140)
gen140<-gen.genealogy(genea140)
gen.rec(gen140, ancestors=c(18305, 18528, 31114, 18325))
```

gen.sibship *Get id numbers of siblings*

Description

Returns the id numbers of the siblings of specified individuals.

Usage

```
gen.sibship( gen, individuals, halfSibling = TRUE, ...)
```

Arguments

gen	An object of class GLgen obtained with gen.genealogy, gen.lineages or gen.branching. Required.
individuals	Vector of individual id numbers. Required.
halfSibling	If halfSibling=TRUE (default) then ids of halfsiblings are also returned. halfSibling=FALSE returns only full sibling ids.
...	Option to pass additionnal arguments automaticaly between methods. Internal use only.

Value

returns a vector of integer

See Also

[gen.genealogy](#) [gen.pro](#) [gen.founder](#) [gen.children](#) [gen.parent](#)

Examples

```
data(geneaJi)
genJi<-gen.genealogy(geneaJi)
gen.sibship(genJi, individuals=21,halfSibling=TRUE)
```

```
data(genea140)
gen140<-gen.genealogy(genea140)
sibs<-gen.sibship(gen140,individuals=10174, halfSibling=FALSE)
gen.graph(gen140, pro=c(10174,sibs))
```

 gen.simuHaplo

Gene dropping simulations - haplotypes

Description

Gene dropping from founders to specified probands. If sequence data is provided the function can translate results to sequences, otherwise will produce proband haplotypes with segments labelled w.r.t. the founder the segment came from

Usage

```
gen.simuHaplo(gen, pro, ancestors, simulNo = 1, RecombRate=c(0,0),
             Reconstruction =0, BP=0, Hapfile=NULL, Mapfile=NULL, seed= 0,
             outDir = getwd())
```

Arguments

gen	An object of class GLgen obtained with gen.genealogy, gen.lineages or gen.branching. Required.
pro	Vector of proband id numbers to be included. Required.
ancestors	Vector of ancestors id numbers to be included
simulNo	Number of simulations to run. Results from extra simulations will be included in the same output text file
RecombRate	The expected number of recombination events in the simulated region per meiosis. E.g.: if simulating a region of 1Centimorgan, recombination rate should be 0.01 Vector of 2 elements, the first is recombination rate for male offspring, the second element specifies rate for female offspring
Reconstruction	Indicator specifying whether the simulation results should be converted into sequence data. Default is Reconstruction = 0, which is no sequence data Reconstruction. If Reconstruction = 1 then user must provide BP, Hapfile, and Mapfile
BP	The length of the simulated region in basepairs. Only required if Reconstruction = 1
Hapfile	The name of a file in the working directory containing sequence data for the founder haplotypes. Each line specifies a haplotype, and should be the same format as the example
Mapfile	The name of a file in the working directory which specifies the locations of the SNPs of Hapfile. This file should contain only one number per each line, each line specifies the position (in BP) of the corresponding SNP in the hapfile
seed	this will seed the random number generator (Mersenne twister) for the simulation. Default is seed = 0 which will choose a random seed. Use the seed for replication purposes
outDir	The directory to write output files to. Default is the current working directory.

Value

A matrix with 3 columns: simulNo, NumMeioses, NumRecomb Will generate output files "Proband_Haplotypes.txt" and "All_nodes_haplotypes.txt" in the specified output directory. If Reconstruction = 1 will also generate "reconstructed_haplotypes.txt"

See Also

[gen.genealogy](#) [gen.simuSample](#) [gen.simuSet](#) [gen.simuSampleFreq](#)

Examples

```
#load the genealogy table
sample_gen_file<-system.file("extdata","sample_gen.csv",package="GENLIB")
sample_gen_table<-read.csv(sample_gen_file)

#convert the genealogy table into a GENLIB object
sample_gen<-gen.genealogy(sample_gen_table, autoComplete = TRUE)

#visualize the genealogy
gen.graph(sample_gen)

probands <- gen.pro(sample_gen)
founders <- gen.founder(sample_gen)

# Simulate the gene dropping from founders to probands
# Generate haplotypes of probands, with segments labelled corresponding to the founder
# If "Reconstruction = 0" as the default so no sequence data is required,
# and the generated haplotypes will be in terms relative position and founder ID

gen.simuHaplo(sample_gen,pro = probands, ancestors= founders, simulNo=1,
              RecombRate=c(0.01,0.02), seed=0, outDir=tempdir())

# The output files will be written to the working directory.
# If you do not wish to give access to this directory,
# then change the working directory with the command setwd() before running the function

#We can also re-run the function with the Reconstruction = 1 option to use sequence data
# this will generate a file of simulated proband haplotypes with sequence data
# But we must provide the function with a Hapfile and Mapfile.

#Hapfile contains the sequence (SNP) data for each founder haplotype.
# Each line should consist of only 2 elements:
# the founder haplotype ID #, and the sequence, seperated by a single whitespace
# the sequence data must be a string and use any characters,
# but must be the same length between all founders.
# Each position in the string corresponds to a BP position in the haplotype,
# so any missing data should be encoded, using a missing data character and not skipped
# Example hapfile:

Hapfile<-system.file("extdata","sequences.txt",package="GENLIB")
```

```

#Mapfile contains the base pair positions of all the SNP's in the Hapfile.
#Each line describes the position of the corresponding character in the sequence
# there should be no empty lines or whitespaces.
# Example mapfile:

Mapfile<-system.file("extdata","SNP_positions.txt",package="GENLIB")

gen.simuHaplo(sample_gen,pro = probands, ancestors= founders, simulNo=2, RecombRate=c(0.01,0.02),
              Reconstruction = 1, BP= 1000000, Hapfile= Hapfile, Mapfile = Mapfile, seed=777,
              outDir=tempdir())

#Since we used the Reconstruction=1 option, we will now get an additional output file
# with sequence data for the proband haplotypes.

```

gen.simuProb

Gene dropping simulations - Probabilities

Description

Returns the probabilities that specified probands inherit disease alleles from ancestors.

Usage

```

gen.simuProb(gen, pro, statePro, ancestors, stateAncestors, simulNo=5000,
             probRecomb=c(0,0), probSurvival=1.0)

```

Arguments

gen	An object of class GLgen obtained with gen.genealogy, gen.lineages or gen.branching. Required.
pro	Vector of proband id numbers to be included. Required.
statePro	Required vector indicating, for each proband in pro, the probability to be calculated: 0 = no disease allele is transmitted 1 = 1 disease allele is transmitted 2 = 2 disease alleles are transmitted 3 = 1 or 2 disease alleles are transmitted
ancestors	Vector of ancestors id numbers to be included. Required.
stateAncestors	Required vector indicating, for each ancestor in ancestors, the genotype state: 0 = no disease allele present 1 = 1 disease allele present 2 = 2 disease alleles present
simulNo	Number of simulations to perform. Default is 5000.
probRecomb	Recombination probabilities for males and females. Default is no recombination.
probSurvival	Survival probability for homozygotes. Default is 1.

Value

A list containing the following:

- the joint probability of specified statePro for all probands,
- the probability of specified statePro for each proband,
- the probability that, 0, 1, ..., and all probands inherit the specified number of disease alleles.

See Also

[gen.genealogy](#) [gen.simuSample](#) [gen.simuSet](#) [gen.simuSampleFreq](#)

Examples

```
data(geneaJi)
genJi<-gen.genealogy(geneaJi)
# Probability that subjects 1 and 29 get 1 and 2 alleles from ancestors 20 and 25,
# that have themselves 2 and 1.
gen.simuProb(genJi, pro=c(1,29), statePro=c(1,2), ancestors=c(20,25), stateAncestors=c(2,1),
             simulNo=10000)
# Probability that subjects 1 and 29 get 1 segment from ancestors 25,
# knowing the segment has a male
# recombination rate of 0.02 and a female recombination rate of 0.04.
gen.simuProb(genJi, pro=c(1,29), statePro=c(1,1), ancestors=c(25), stateAncestors=c(1),
             simulNo=10000, probRecomb = c(0.02, 0.04))
# Probability that subjects 1 and 29 get 1 and 2 alleles from ancestors 20 and 25,
# that have themselves 2 and 1 and knowing that homozygous people have a survival rate
# of 0.50.
gen.simuProb(genJi, pro=c(1,29), statePro=c(1,2), ancestors=c(20,25), stateAncestors=c(2,1),
             simulNo=10000, probSurvival=0.5)
```

gen.simuSample

Gene dropping simulations - Sample

Description

Returns the number of alleles transmitted to specified probands from ancestors

Usage

```
gen.simuSample(gen, pro, ancestors, stateAncestors, simulNo = 5000)
```

Arguments

gen	An object of class GLgen obtained with gen.genealogy, gen.lineages or gen.branching. Required.
pro	Vector of proband id numbers to be included. Required.
ancestors	Vector of ancestors id numbers to be included. Required.

`stateAncestors` Required vector indicating, for each ancestor in `ancestors`, the genotype state:
 0 = no disease allele present
 1 = 1 disease allele present
 2 = 2 disease alleles present

`simulNo` Number of simulations to perform. Default is 5000.

Value

A matrix with number of columns equal to the number of simulations and number of rows equal to the number of probands.

See Also

[gen.genealogy](#) [gen.simuProb](#) [gen.simuSet](#) [gen.simuSampleFreq](#)

Examples

```
data(geneaJi)
genJi<-gen.genealogy(geneaJi)
simu_1000<-gen.simuSample(genJi, pro=c(1,29), ancestors=c(20,25), stateAncestors=c(2,1),
                          simulNo=1000)
# Number of alleles received by probands
table(simu_1000)
# Number of alleles received by simulation
table(colSums(simu_1000))
# Number of alleles received by each probant
table(simu_1000[1,],simu_1000[2,])

data(geneaJi)
genJi<-gen.genealogy(geneaJi)
simu_5000<-gen.simuSample(genJi, pro=c(1,29), ancestors=c(20,25), stateAncestors=c(2,1),
                          simulNo=5000)
# Number of alleles received by probands
table(simu_5000)
# Number of alleles received by simulation
table(colSums(simu_5000))
# Number of alleles received by each probant
table(simu_5000[1,],simu_5000[2,])
```

`gen.simuSampleFreq` *Gene dropping simulations - Frequencies*

Description

Returns the number of alleles transmitted to specified probands from ancestors

Usage

```
gen.simuSampleFreq( gen, pro, ancestors, stateAncestors, simulNo = "5000")
```

Arguments

gen	An object of class GLgen obtained with gen.genealogy, gen.lineages or gen.branching. Required.
pro	Vector of proband id numbers to be included. Required.
ancestors	Vector of ancestors id numbers to be included. Required.
stateAncestors	Required vector indicating, for each ancestor in ancestors, the genotype state: 0 = no disease allele present 1 = 1 disease allele present 2 = 2 disease alleles present
simulNo	Number of simulations to perform. Default is 5000.

Value

A data frame containing for each row (one row per proband): **Alleles.transmitted.0 Alleles.transmitted.1 Alleles.transmitted.2**

See Also

[gen.genealogy](#) [gen.simuProb](#) [gen.simuSet](#) [gen.simuSample](#)

Examples

```
data(geneaJi)
genJi<-gen.genealogy(geneaJi)
gen.simuSampleFreq(genJi, pro=c(1,29), ancestors=c(20,25),stateAncestors=c(2,1), simulNo = 1000)
```

gen.simuSet

Gene dropping simulations with specified transmission probabilities

Description

Returns the number of alleles transmitted to specified probands from ancestors considering specified transmission probabilities.

Usage

```
gen.simuSet(gen, pro, ancestors, stateAncestors,
            probMatrix=matrix(c(
                c(1,0.5,0,0.5,0.25,0,0,0,0,1,1,1,1,0.75,0.5,1,0.5,0),
                c(1,0.5,0,0.5,0.25,0,0,0,0,1,1,1,1,0.75,0.5,1,0.5,0)),
                nrow=3, ncol=12),
            simulNo = 5000)
```

Arguments

gen	An object of class GLgen obtained with gen.genealogy, gen.lineages or gen.branching. Required.
pro	Vector of proband id numbers to be included. Required.
ancestors	Vector of ancestors id numbers to be included. Required.
stateAncestors	Required vector indicating, for each ancestor in ancestors, the genotype state: 0 = no disease allele present 1 = 1 disease allele present 2 = 2 disease alleles present
probMatrix	Matrix of transmission probabilities of 0, 1, or 2 disease alleles. Default is: [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [1,] 1.0 0.50 0 1 1.00 1.0 1.0 0.50 0 1 1.00 1.0 [2,] 0.5 0.25 0 1 0.75 0.5 0.5 0.25 0 1 0.75 0.5 [3,] 0.0 0.00 0 1 0.50 0.0 0.0 0.00 0 1 0.50 0.0 where rows are genotype states (0,1,2) for the father and columns are states for the mother (in blocks of 3). 1st block (col 1-3) are the probabilities of transmitting 0 allele to a son. 2nd block (col 4-6) are the probabilities of transmitting 1 allele to a son. 3rd block (col 7-9) are the probabilities of transmitting 0 allele to a daughter. 4th block (col 10-12) are the probabilities of transmitting 1 allele to a daughter.
simulNo	Number of simulations to perform. Default is 5000.

Value

A matrix with number of columns equal to the number of simulations and number of rows equal to the number of probands.

See Also

[gen.genealogy](#) [gen.simuProb](#) [gen.simuSample](#)

Examples

```
# A case where only male subjects can receive alleles
data(geneaJi)
genJi<-gen.genealogy(geneaJi)
onlyThroughMale<-matrix(c(c(1,0.5,0,0.5,0.25,0,0,0,0,1,1,1,1,0.75,0.5,1,0.5,0), rep(1,18)),
  nrow=3, ncol=12)
gen.graph(genJi, indVarAffected=c(28,27,25,17,12,8,4,1), varAffected=c(28,27,25,17,12,8,4,1))
simu_1000a<-gen.simuSet(genJi, pro=c(1,28), ancestors=c(17,25), stateAncestors=c(1,1),
  simulNo = 1000,probMatrix=onlyThroughMale)
# Number of alleles received by probands
table(simu_1000a)
# Number of alleles received by all probands at each simulation
table(colSums(simu_1000a))
# Number of alleles received by each probant
```

```

table(simu_1000a[1,],simu_1000a[2,])

# A case where subjects are limited to one copy compared to what is normally expected
max1Allreceived<-matrix(rep(c(1,0.5,0,0.5,0.25,0,0,0,0,1,1,1,1,1,1,1,1), 2) , nrow=3, ncol=12)
gen.graph(genJi)
simu_1000b<-gen.simuSet(genJi, pro=c(1,29), ancestors=25, stateAncestors=1, simulNo=10000,
                        probMatrix=max1Allreceived)
# Normal case matrix not changed
simu_1000original<-gen.simuSet(genJi, pro=c(1,29), ancestors=25, stateAncestors=1, simulNo=10000)
# Number of alleles received by each probant
table(simu_1000b)
table(simu_1000original)
# Number of alleles received by all probants at each simulation
table(colSums(simu_1000b))
table(colSums(simu_1000original))
# Number of alleles received by each probant
table(simu_1000b[1,],simu_1000b[2,])
table(simu_1000original[1,],simu_1000original[2,])

```

genea140

Genealogical information for 140 individuals from the Quebec Reference Sample

Description

A genealogical corpus made of 41523 individuals from the province of Quebec, Canada. A total of 140 individuals have been sampled in seven sub-populations, listed in pop140, and their genealogies were reconstructed as far back as possible using the BALSAC population register and the Early Quebec Population Register.

Usage

```
data(gene140)
```

Format

A data frame with 41523 observations on 11 variables.

[,1] ind: An individual's ID number
[,2] father: This individual's father ID number
[,3] mother: This individual's mother ID number
[,4] sex: Individual's sex coded 1/2 for male/female

Source

Balsac

References

<http://www.quebecgenpop.ca>

Roy-Gagnon, M.-H., Moreau, C., Bherer, C., St-Onge, P., Sinnett, D., Laprise, C., Vezina, H., Labuda, D. (2011). Genomic and genealogical investigation of the French Canadian founder population structure. *Human Genetics*, 129(5), 521-31.

geneaJi

Highly inbred pedigree

Description

A modified version of a pedigree of two Jicaque Indians studied by Chapman & Jacquard (1971).

Usage

`data(geneaJi)`

Format

A data frame with 29 observations on 4 variables.

[, 1] ind An individual's ID number

[, 2] father This individual's father ID number

[, 3] mother This individual's mother ID number

[, 4] sex Individual's sex coded 1/2 for male/female

Source

CHAPMAN & JACQUART (1971)

References

Chapman, A.M., & Jacquard, A. (1971). Un isolat d'Amérique centrale: Les indiens Jicaques du Honduras. Paris: Presses universitaires de France.

 GLgen-class

 Class "GLgen"

Description

Object containing a genealogy.

Objects from the Class

Objects can be created by calls of the form `gen.genealogy('data.frame')`.

Extends

Class "GLgroup", directly.

Methods

initialize signature(.Data="integer",Date="character"): ...

depth signature(x = "GLgen") Gives the depth of the genealogy.

length signature(x = "GLgen") Gives the number of individuals in the genealogy.

Examples

```
showClass("GLgen")
```

 GLgroup-class

 Class "GLgroup"

Description

Object representing a set of proband in different groups.

Each element of this list is a group named with the proband number.

Objects from the Class

Objects can be created by calls of the form `new("GLgroup", 'list')`.

Extends

Class "GLgen", directly.

Methods

initialize signature(.Data=list): ...

[signature(object = "GLgroup", ANY, ANY, ANY):...

Examples

```
showClass("GLgroup")
```

```
GLmultiList-class      Class "GLmultiList"
```

Description

Objects created to carry information mostly between invisible functions.

Objects from the Class

Objects can be created by calls of the form `new("GLmultiList", 'Array')`.

Slots

.Data: Object of class "list" ~~

liste: Object of class "list" ~~

Extends

Class "[list](#)", from data part. Class "[vector](#)", by class "list", distance 2.

Methods

No methods defined with class "GLmultiList" in the signature.

Examples

```
showClass("GLmultiList")
```

```
pop140                Population of origin of the 140 Quebec samples
```

Description

The 140 individuals from the genealogical corpus from Quebec were sampled from 7 different populations from 5 regions; Quebec City, Montreal, Saguenay, North Shore, Gaspesia. In Gaspesia we find 3 different populations: French-Canadians, Acadians and Loyalists.

Usage

```
data(pop140)
```


Format

, 1 ind: An individual's ID number

[, 2] pop: This individual's population

Source

Balsac

References

<http://www.quebecgenpop.ca/home.html>

Roy-Gagnon, M.-H., Moreau, C., Bherer, C., St-Onge, P., Sinnett, D., Laprise, C., Vezina, H., Labuda, D. (2011). Genomic and genealogical investigation of the French Canadian founder population structure. *Human Genetics*, 129(5), 521-31.

Index

- * **Utilities**
 - gen.nomen, 36
- * **aplot**
 - gen.graph, 23
- * **attribute**
 - gen.depth, 11
- * **classes**
 - Classes of basic handling of genealogy data, 3
 - Classes of GLgroup handling, 4
 - GLgen-class, 55
 - GLgroup-class, 55
 - GLmultiList-class, 56
- * **datagen**
 - gen.simuProb, 48
 - gen.simuSample, 49
 - gen.simuSampleFreq, 50
 - gen.simuSet, 51
- * **datasets**
 - genea140, 53
 - geneaJi, 54
 - pop140, 56
- * **manip**
 - gen.branching, 6
 - gen.children, 7
 - gen.completeness, 8
 - gen.completenessVar, 9
 - gen.f, 11
 - gen.fCI, 13
 - gen.find.Min.Distance.MRCA, 14
 - gen.findDistance, 15
 - gen.findFounders, 16
 - gen.findMRCA, 17
 - gen.founder, 18
 - gen.gc, 19
 - gen.genealogy, 20
 - gen.genout, 21
 - gen.half.founder, 25
 - gen.implex, 26
 - gen.implexVar, 27
 - gen.lineages, 28
 - gen.max, 29
 - gen.mean, 30
 - gen.meangendepth, 31
 - gen.meangendepthVar, 32
 - gen.min, 33
 - gen.nochildren, 34
 - gen.occ, 37
 - gen.parent, 38
 - gen.phi, 39
 - gen.phiCI, 40
 - gen.phiMean, 41
 - gen.phiOver, 42
 - gen.pro, 43
 - gen.rec, 44
 - gen.sibship, 45
- * **methods**
 - gen.climbPAR, 8
 - gen.getAncestorsPAR, 22
 - gen.getFoundersPAR, 23
- * **package**
 - GenlibR-package, 3
- * **utilities**
 - gen.noind, 35
 - gen.nowomen, 36
- [,GLCGMatrixGroupSingle,ANY,ANY,ANY-method
(Classes of GLgroup handling),
4
- [,GLgroup,ANY,ANY,ANY-method
(GLgroup-class), 55
- [,GLmultiArray4,ANY,ANY,ANY-method
(Classes of basic handling of
genealogy data), 3
- [,GLmultiFGroup,ANY,ANY,ANY-method
(Classes of GLgroup handling),
4
- [,GLmultiFGroupSingle,ANY,ANY,ANY-method
(Classes of GLgroup handling),

- 4
- [,GLmultiMatrix,ANY,ANY,ANY-method
(Classes of basic handling of
genealogy data), 3
- [,GLmultiNumber,ANY,ANY,ANY-method
(Classes of basic handling of
genealogy data), 3
- [,GLmultiPhiGroup,ANY,ANY,ANY-method
(Classes of GLgroup handling),
4
- [,GLmultiPhiGroupSingle,ANY,ANY,ANY-method
(Classes of GLgroup handling),
4
- [,GLmultiVector,ANY,ANY,ANY-method
(Classes of basic handling of
genealogy data), 3
- [<-,GLCGMatrixGroupSingle,ANY,ANY,ANY-method
(Classes of GLgroup handling),
4
- [<-,GLmultiArray4,ANY,ANY,ANY-method
(Classes of basic handling of
genealogy data), 3
- [<-,GLmultiFGroup,ANY,ANY,ANY-method
(Classes of GLgroup handling),
4
- [<-,GLmultiFGroupSingle,ANY,ANY,ANY-method
(Classes of GLgroup handling),
4
- [<-,GLmultiMatrix,ANY,ANY,ANY-method
(Classes of basic handling of
genealogy data), 3
- [<-,GLmultiNumber,ANY,ANY,ANY-method
(Classes of basic handling of
genealogy data), 3
- [<-,GLmultiPhiGroup,ANY,ANY,ANY-method
(Classes of GLgroup handling),
4
- [<-,GLmultiPhiGroupSingle,ANY,ANY,ANY-method
(Classes of GLgroup handling),
4
- [<-,GLmultiVector,ANY,ANY,ANY-method
(Classes of basic handling of
genealogy data), 3
- array, 4, 5
- Classes of basic handling of genealogy
data, 3
- Classes of GLgroup handling, 4
- depth,GLgen-method (GLgen-class), 55
- depth,GLmultiArray4-method (Classes of
basic handling of genealogy
data), 3
- depth,GLmultiMatrix-method (Classes of
basic handling of genealogy
data), 3
- depth,GLmultiNumber-method (Classes of
basic handling of genealogy
data), 3
- depth,GLmultiVector-method (Classes of
basic handling of genealogy
data), 3
- Dim,GLmultiArray4-method (Classes of
basic handling of genealogy
data), 3
- Dim,GLmultiFGroup-method (Classes of
GLgroup handling), 4
- Dim,GLmultiFGroupSingle-method
(Classes of GLgroup handling),
4
- Dim,GLmultiMatrix-method (Classes of
basic handling of genealogy
data), 3
- Dim,GLmultiNumber-method (Classes of
basic handling of genealogy
data), 3
- Dim,GLmultiPhiGroup-method (Classes of
GLgroup handling), 4
- Dim,GLmultiPhiGroupSingle-method
(Classes of GLgroup handling),
4
- Dim,GLmultiVector-method (Classes of
basic handling of genealogy
data), 3
- gen.branching, 6, 21, 24, 29
- gen.children, 7, 35, 38, 43, 45
- gen.climbPAR, 8, 22
- gen.completeness, 8, 19, 26, 31
- gen.completenessVar, 9
- gen.depth, 11, 29, 30, 34
- gen.f, 11, 13, 39
- gen.fCI, 13
- gen.find.Min.Distance.MRCA, 14, 15–17
- gen.findDistance, 14, 15, 16, 17, 23
- gen.findFounders, 14, 15, 16, 17, 23
- gen.findMRCA, 8, 14–16, 17, 22
- gen.founder, 7, 14–17, 18, 25, 38, 43, 45

- gen.gc, [10](#), [19](#), [26](#), [28](#), [31](#), [33](#), [38](#), [44](#)
- gen.genealogy, [6](#), [7](#), [9](#), [11](#), [12](#), [14–19](#), [20](#), [21](#), [24–26](#), [29–31](#), [34–39](#), [41–45](#), [47](#), [49–52](#)
- gen.genout, [21](#), [21](#), [24](#), [29](#)
- gen.getAncestorsPAR, [8](#), [22](#)
- gen.getFoundersPAR, [23](#)
- gen.graph, [6](#), [21](#), [23](#), [29](#)
- gen.half.founder, [18](#), [25](#), [43](#)
- gen.implex, [9](#), [19](#), [26](#), [31](#), [38](#), [44](#)
- gen.implexVar, [27](#)
- gen.lineages, [6](#), [21](#), [28](#)
- gen.max, [29](#), [30](#), [34](#)
- gen.mean, [11](#), [29](#), [30](#), [34](#)
- gen.meangendepth, [9](#), [11](#), [19](#), [26](#), [29](#), [30](#), [31](#), [34](#), [38](#), [44](#)
- gen.meangendepthVar, [32](#)
- gen.min, [11](#), [29](#), [30](#), [33](#)
- gen.nochildren, [34](#), [35–37](#)
- gen.noind, [35](#), [35](#), [36](#), [37](#)
- gen.nomen, [35](#), [36](#), [37](#)
- gen.nowomen, [35](#), [36](#), [36](#)
- gen.occ, [9](#), [19](#), [26](#), [31](#), [37](#), [44](#)
- gen.parent, [7](#), [18](#), [25](#), [38](#), [43](#), [45](#)
- gen.phi, [12](#), [39](#), [41](#), [42](#)
- gen.phiCI, [40](#), [41](#), [42](#)
- gen.phiMean, [41](#), [41](#), [42](#)
- gen.phiOver, [41](#), [42](#)
- gen.pro, [7](#), [18](#), [25](#), [38](#), [43](#), [45](#)
- gen.rec, [9](#), [19](#), [26](#), [31](#), [38](#), [44](#)
- gen.sibship, [7](#), [38](#), [45](#)
- gen.simuHaplo, [46](#)
- gen.simuProb, [48](#), [50–52](#)
- gen.simuSample, [47](#), [49](#), [49](#), [51](#), [52](#)
- gen.simuSampleFreq, [47](#), [49](#), [50](#), [50](#)
- gen.simuSet, [47](#), [49–51](#), [51](#)
- genea140, [53](#)
- geneaJi, [54](#)
- GenlibR-package, [3](#)
- GLCGMatrixGroupSingle-class (Classes of GLgroup handling), [4](#)
- GLgen, [55](#)
- GLgen-class, [55](#)
- GLgroup, [55](#)
- GLgroup-class, [55](#)
- GLmultiArray4-class (Classes of basic handling of genealogy data), [3](#)
- GLmultiFGroup-class (Classes of GLgroup handling), [4](#)
- GLmultiFGroupSingle-class (Classes of GLgroup handling), [4](#)
- GLmultiList-class, [56](#)
- GLmultiMatrix-class (Classes of basic handling of genealogy data), [3](#)
- GLmultiNumber-class (Classes of basic handling of genealogy data), [3](#)
- GLmultiPhiGroup-class (Classes of GLgroup handling), [4](#)
- GLmultiPhiGroupSingle-class (Classes of GLgroup handling), [4](#)
- GLmultiVector-class (Classes of basic handling of genealogy data), [3](#)
- initialize, GLgen-method (GLgen-class), [55](#)
- initialize, GLgroup-method (GLgroup-class), [55](#)
- length, GLgen-method (GLgen-class), [55](#)
- list, [56](#)
- matrix, [4](#), [5](#)
- pop140, [56](#)
- structure, [4](#), [5](#)
- vector, [4](#), [5](#), [56](#)