Package 'LCAextend'

October 12, 2022

Type Package

Title Latent Class Analysis (LCA) with Familial Dependence in Extended Pedigrees

Version 1.3

Date 2018-07-05

Author Arafat TAYEB <arafat.tayeb@ircm.qc.ca>, Alexandre BUREAU <alexandre.bureau@msp.ulaval.ca> and Aurelie Labbe <aurelie.labbe@mcgill.ca>

Maintainer Alexandre BUREAU <alexandre.bureau@msp.ulaval.ca>

Depends R (>= 2.1.0)

Imports boot, mvtnorm, rms, kinship2

Description Latent Class Analysis of phenotypic measurements in pedigrees and model selection based on one of two methods: likelihood-based cross-validation and Bayesian Information Criterion. Computation of individual and triplet child-parents weights in a pedigree is performed using an upward-downward algorithm. The model takes into account the familial dependence defined by the pedigree structure by considering that a class of a child depends on his parents classes via triplet-transition probabilities of the classes. The package handles the case where measurements are available on all subjects and the case where measurements are available only on symptomatic (i.e. affected) subjects. Distributions for discrete (or ordinal) and continuous data are currently implemented. The package can deal with missing data.

License GPL

LazyLoad yes

URL https://CRAN.R-project.org/package=LCAextend

Repository CRAN

NeedsCompilation no

Date/Publication 2018-07-07 15:40:21 UTC

R topics documented:

alpha.compute . . . 2
attrib.dens . . . 3
dens.norm . . . 4
dens.prod.ordi . . . 5
downward . . . 6
downward.connect . . . 8
e.step . . . 10
init.norm . . . 12
init.ordi . . . 13
init.p.trans . . . 14
lca.model . . . 15
model.select . . . 18
n.param . . . 21
optim.const.ordi . . . 22
optim.diff.norm . . . 23
optim.equal.norm . . . 25
optim.gene.norm . . . 26
optim.indep.norm . . . 27
optim.noconst.ordi . . . 28
optim.probs . . . 29
p.compute . . . 30
p.post.child . . . 31
p.post.found . . . 32
param.cont . . . 33
param.ordi . . . 34
ped.cont . . . 35
ped.ordi . . . 35
peel . . . 36
probs . . . 36
upward . . . 37
upward.connect . . . 39
weight.famdep . . . 41
weight.nuc . . . 43

Index **46**

`alpha.compute` *computes cumulative logistic coefficients using probabilities*

Description

computes cumulative logistic coefficients using probabilities.

Usage

```
alpha.compute(p)
```

Arguments

p	a vector of probabilities (positive entries summing to 1).

Details

If p has one value (equal to 1) alpha.compute returns NA, if it has S (S>=2) values, alpha.compute returns S-1 coefficients alpha such that if Y is a random variable taking values in {1,...,S} with probabilities p, coefficients alpha[i] are given by:

$$p_1 + ... + p_i = P(Y \leq i) = \frac{\exp(\alpha_1 + ... + \alpha_i)}{1 + \exp(\alpha_1 + ... + \alpha_i)},$$

for all i<=S-1.

Value

The function returns alpha: a vector of S-1 cumulative logistic coefficients.

See Also

alpha.compute is the inverse function of p.compute

Examples

```
# a vector of probability
p <- c(0,0.2,0,0,0.3,0.4,0.1,0,0)
alpha.compute(p)
#gives -Inf -1.38 0 0 1.38 0 2.19 Inf Inf
p.compute(alpha.compute(rep(1/5,5)))
```

---

attrib.dens *associates to a function of density parameter optimization an attribute to distinguish between ordinal and normal cases*

Description

associates to a function of density parameter optimization an attribute to distinguish between ordinal and normal cases. This is an internal function not meant to be called by the user.

Usage

```
attrib.dens(optim.param)
```

Arguments

optim.param	the function used to estimate the parameters of the measurements.

Details

Available optim.param functions are optim.noconst.ordi, optim.const.ordi for ordinal measurements and optim.indep.norm, optim.diff.norm, optim.equal.norm and optim.gene.norm for multinormal measurements. The attribution uses the internal function attr and the attribute name used is type. The user can make his own optima.param function and has to associate an attribute type to it to be used instead of the available ones.

Value

The function returns the same optim.param with an attribute type taking values in ordi or norm.

Examples

```
optim.param <- optim.indep.norm
optim.param <- attrib.dens(optim.param)
```

---

dens.norm *computes the multinormal density of a given continuous measurement vector for all classes*

Description

computes the density of an individual's continuous measurement vector for all latent classes, eventually taking covariates into account. This is an internal function not meant to be called by the user.

Usage

```
dens.norm(y.x, param, var.list = NULL)
```

Arguments

y.x	a vector y of values of the measurement followed by the values x of covariates, if any,
param	a list of the multinormal density parameters: means mu and variances-covariances sigma,
var.list	a list of integers indicating which covariates (taken from x) are used for a given type of measurement.

Details

For each class k, the function computes the multinormal density with means param$mu[[k]] and variances-covariances matrix param$sigma[[k]] for the individual's measurement vector. Treatment of covariates is not yet implemented, and any provided covariate value will be ignored.

Value

The function returns a vector dens of length K, where dens[k] is the density of the measurements if the individual belongs to class k.

Examples

```
#data
data(ped.cont)
status <- ped.cont[,6]
y <- ped.cont[status==2,7:ncol(ped.cont)]
#param
data(param.cont)
#the function applied for measurement of the first individual in the ped.ordi
dens.norm(y.x=y[1,],param.cont)
```

---

dens.prod.ordi *computes the probability of a given discrete measurement vector for all classes under a product of multinomial*

Description

computes the probability of an individual's discrete measurement vector for all latent classes under a multinomial distribution product, eventually taking covariates into account. This is an internal function not meant to be called by the user.

Usage

```
dens.prod.ordi(y.x, param, var.list = NULL)
```

Arguments

y.x	a vector y of values of the ordinal variables (measurements) followed by the values x of covariates, if any,
param	a list of the parameters alpha (cumulative logistic coefficients), see init.ordi,
var.list	a list of integers indicating which covariates (taken from x) are used for a given type of measurement.

Details

If there are K latent classes, d measurements and each measurement has S[j] possible values, alpha is a list of d elements, each is a K times S[j]+length{var.list[[j]]} matrix. For a class C=k, dens[k]=$\prod_{j=1}^{d} P(Y_j = y_j | C = k, X_j = x_j)$, where $P(Y_j = y_j | C = k, X_j = x_j)$ is computed from the cumulative logistic coefficients alpha[[j]][k,] and covariates x[var.list[[j]]],

Value

The function returns a vector dens of length K, where dens[k] is the probability of measurement vector y with covariates x, if the individual belongs to class k.

See Also

See Also init.ordi,

Examples

```
#data
data(ped.ordi)
status <- ped.ordi[,6]
y <- ped.ordi[status==2,7:ncol(ped.ordi)]
#param
data(param.ordi)
#the function applied for measurement of the first individual in the ped.ordi
dens.prod.ordi(y.x=y[1,],param.ordi)
```

---

downward *performs the downward step of the peeling algorithm and computes unnormalized triplet and individual weights*

Description

computes the probability of measurements above connectors and their classes given the model parameters, and returns the unnormalized triplet and individual weights. This is an internal function not meant to be called by the user.

Usage

```
downward(id, dad, mom, status, probs, fyc, peel, res.upward)
```

Arguments

id	individual ID of the pedigree,
dad	dad ID,
mom	mom ID,
status	symptom status: (2: symptomatic, 1: without symptoms, 0: missing),
probs	a list of probability parameters of the model,
fyc	a matrix of n times K+1 given the density of observations of each individual if allocated to class k, where n is the number of individuals and K is the total number of latent classes in the model,
peel	a list of pedigree peeling containing connectors by peeling order and couples of parents,
res.upward	result of the upward step of the peeling algorithm, see upward.

Details

This function computes the probability of observations above connectors and their classes using the function `downward.connect`, for each connector, if `Y_above(i)` is the observations above connector `i` and `S_i` and `C_i` are his status and his class respectively, the functions computes `P(Y_above(i),S_i,C_i)` by computing a downward step for the parent of connector `i` who is also a connector. These quantities are used by the function `weight.nuc` to compute the unnormalized triplet weights `ww` and the unnormalized individual weights `w`.

Value

The function returns a list of 2 elements:

`ww`	unnormalized triplet weights, an array of `n` times 2 times `K+1` times `K+1` times `K+1`, where `n` is the number of individulas and `K` is the total number of latent classes in the model, see `e.step` for more details,
`w`	unnormalized individual weights, an array of `n` times 2 times `K+1`, see `e.step`.

References

TAYEB et al.: Solving Genetic Heterogeneity in Extended Families by Identifying Sub-types of Complex Diseases. Computational Statistics, 2011, DOI: 10.1007/s00180-010-0224-2.

See Also

See also `downward.connect`.

Examples

```
#data
data(ped.cont)
data(peel)
fam <- ped.cont[,1]
id <- ped.cont[fam==1,2]
dad <- ped.cont[fam==1,3]
mom <- ped.cont[fam==1,4]
status <- ped.cont[fam==1,6]
y <- ped.cont[fam==1,7:ncol(ped.cont)]
peel <- peel[[1]]
#standardize id to be 1, 2, 3, ...
id.origin <- id
standard <- function(vec) ifelse(vec%in%id.origin,which(id.origin==vec),0)
id <- apply(t(id),2,standard)
dad <- apply(t(dad),2,standard)
mom <- apply(t(mom),2,standard)
peel$couple <- cbind(apply(t(peel$couple[,1]),2,standard),
                     apply(t(peel$couple[,2]),2,standard))
for(generat in 1:peel$generation)
    peel$peel.connect[generat,] <- apply(t(peel$peel.connect[generat,]),2,standard)
#probs and param
data(probs)
data(param.cont)
```

```
#densities of the observations
fyc <- matrix(1,nrow=length(id),ncol=length(probs$p)+1)
fyc[status==2,1:length(probs$p)] <- t(apply(y[status==2,],1,dens.norm,param.cont,NULL))
#the upward step
res.upward <- upward(id,dad,mom,status,probs,fyc,peel)
#the function
downward(id,dad,mom,status,probs,fyc,peel,res.upward)
```

`downward.connect` *performs a downward step for a connector*

Description

computes the probability of the measurements above a connector and the connector latent class given the model parameters. This is an internal function not meant to be called by the user.

Usage

```
downward.connect(connect, parent1, parent2, bro.connect, status,
                 probs, fyc, p.ybarF.c, res.upward)
```

Arguments

`connect`	a connector in the pedigree (individual with parents and children in the pedigree),
`parent1`	one of the connector parent who is also a connector,
`parent2`	the other parent of the connector (not a connector),
`bro.connect`	siblings of the connector,
`status`	a vector of symptom status,
`probs`	a list of all probability parameters of the model,
`fyc`	a matrix of `n` times `K+1` given the density of observations of each individual if allocated to class `k`, where `n` is the number of individuals and `K` is the total number of latent classes in the model. the `K+1` corresponds to the unaffected class,
`p.ybarF.c`	a array of dimension `n` times 2 times `K+1` giving the probability of observations above the individual, depending on his status and his class and conditionally to his class,
`res.upward`	the result of the upward step of the peeling algorithm, see `upward`.

Details

If `Y_above(i)` is the measurements above connector `i` and `S_i` and `C_i` are his status and his class respectively, the function computes `P(Y_above(i),S_i,C_i)` by computing a downward step for the parent of connector `i` who is also a connector.

Value

The function returns `p.ybarF.c` updated for connector `i`.

References

TAYEB et al.: Solving Genetic Heterogeneity in Extended Families by Identifying Sub-types of Complex Diseases. Computational Statistics, 2011, DOI: 10.1007/s00180-010-0224-2.

See Also

See also `downward`

Examples

```
#data
data(ped.cont)
data(peel)
fam <- ped.cont[,1]
id <- ped.cont[fam==1,2]
dad <- ped.cont[fam==1,3]
mom <- ped.cont[fam==1,4]
status <- ped.cont[fam==1,6]
y <- ped.cont[fam==1,7:ncol(ped.cont)]
peel <- peel[[1]]
#standardize id to be 1, 2, 3, ...
id.origin <- id
standard <- function(vec) ifelse(vec%in%id.origin,which(id.origin==vec),0)
id <- apply(t(id),2,standard)
dad <- apply(t(dad),2,standard)
mom <- apply(t(mom),2,standard)
peel$couple <- cbind(apply(t(peel$couple[,1]),2,standard),
                     apply(t(peel$couple[,2]),2,standard))
for(generat in 1:peel$generation)
peel$peel.connect[generat,] <- apply(t(peel$peel.connect[generat,]),2,standard)
#the 2nd connector
generat <- peel$generation-1
connect <- peel$peel.connect[generat,]
connect <- connect[connect>0][1]
parent1.connect <- intersect(peel$peel.connect[generat+1,],c(dad[id==connect],
                                                             mom[id==connect]))
parent2.connect <- setdiff(c(dad[id==connect],mom[id==connect]),parent1.connect)
bro.connect <- union(id[dad==parent1.connect],id[mom==parent1.connect])
bro.connect <- setdiff(bro.connect,connect)
#probs and param
data(probs)
data(param.cont)
#densities of the observations
fyc <- matrix(1,nrow=length(id),ncol=length(probs$p)+1)
fyc[status==2,1:length(probs$p)] <- t(apply(y[status==2,],1,dens.norm,param.cont,NULL))
#probability of the observations below
p.ybarF.c <- array(1,dim=c(length(id),2,length(probs$p)+1))
#the upward step
```

```
res.upward <- upward(id,dad,mom,status,probs,fyc,peel)
#the function
downward.connect(connect,parent1.connect,parent2.connect,bro.connect,status,
                 probs,fyc,p.ybarF.c,res.upward)
```

`e.step` *performs the E step of the EM algorithm for a single pedigree for both cases with and without familial dependence*

Description

computes triplet and individual weights the E step of the EM algorithm for all pedigrees in the data, in both cases with and without familial dependence. This is an internal function not meant to be called by the user.

Usage

```
e.step(ped, probs, param, dens, peel, x = NULL, var.list = NULL,
       famdep = TRUE)
```

Arguments

`ped`	a matrix representing pedigrees and measurements: `ped[,1]` family ID, `ped[,2]` subjects ID, `ped[,3]` dad ID, `ped[,4]` mom ID, `ped[,5]` sex, `ped[,6]` symptom status: (2: symptomatic, 1: without symptoms, 0: missing), `ped[,7:ncol(ped)]` measurements, each column corresponds to a phenotypic measurement,
`probs`	a list of probability parameters of the model, see below for more details,
`param`	a list of measurement distribution parameters of the model, see below for more details,
`dens`	distribution of the mesurements, used in the model (multinormal, multinomial,...)
`peel`	a list of pedigree peeling containing connectors by peeling order and couples of parents,
`x`	covariates, if any. Default is `NULL`,
`var.list`	a list of integers indicating which covariates (taken from `x`) are used for a given type of measurement. Default is `NULL`,
`famdep`	a logical variable indicating if familial dependence model is used or not. Default is `TRUE`. In models without familial dependence, individuals are treated as independent and pedigree structure is meaningless. In models with familial dependence, a child class depends in his parents classes via a triplet-transition probability,

Details

probs is a list of initial probability parameters:

For models with familial dependence:

p a probability vector, each p[c] is the probability that an symptomatic founder is in class c for c>=1,

p0 the probability that a founder without symptoms is in class 0,

p.trans an array of dimension K times K+1 times K+1, where K is the number of latent classes of the model, and is such that p.trans[c_i,c_1,c_2] is the conditional probability that a symptomatic individual i is in class c_i given that his parents are in classes c_1 and c_2,

p0connect a vector of length K, where p0connect[c] is the probability that a connector without symptoms is in class 0, given that one of his parents is in class c>=1 and the other in class 0,

p.found the probability that a founder is symptomatic,

p.child the probability that a child is symptomatic,

For models without familial dependence, all individuals are independent:

p a probability vector, each p[c] is the probability that an symptomatic individual is in class c for c>=1,

p0 the probability that an individual without symptoms is in class 0,

p.aff the probability that an individual is symptomatic,

param is a list of measurement density parameters: the coefficients alpha (cumulative logistic coefficients see alpha.compute) in the case of discrete or ordinal data, and means mu and variances-covariances matrices sigma in the case of continuous data,

Value

The function returns a list of 3 elements:

ww	triplet posterior probabilities, an array of n (the number of individuals) times 2 times K+1 times K+1 times K+1, where K is the total number of latent classes of the model. For an individual i, the triplet probability ww[i,s,c,c_1,c_2] is the posterior probability that individual i belongs to class c when his symptom status is s and given that his parents classes are c_1 and c_2, where s takes two values 1 for affected and 2 for unaffected. In particular, all ww[,2,,,] are zeros for affected individuals and all ww[,1,,,] are zeros for unaffected individuals. For missing individuals (unkown symptom status), both ww[,1,,,] and ww[,2,,,] are full,
w	individual posterior probabilities, an array of n times 2 times K+1 , where n is the number of individuals and is such that w[i,s,c] is the posterior probability that individual i belongs to class c when his symptom status is s, where s takes two values 1 for affected and 2 for unaffected. In particular, all w[,2,] are zeros for affected individuals and all w[,1,] are zeros for unaffected individuals. For missing individuals (unkown symptom status), both w[,1,] and w[,2,] are full,
ll	log-likelihood of the considered model and parameters.

References

TAYEB et al.: Solving Genetic Heterogeneity in Extended Families by Identifying Sub-types of Complex Diseases. Computational Statistics, 2011, DOI: 10.1007/s00180-010-0224-2.

See Also

See also `weight.famdep`, `lca.model`.

Examples

```
#data
data(ped.cont)
data(peel)
#probs and probs
data(probs)
data(param.cont)
#the function
e.step(ped.cont,probs,param.cont,dens.norm,peel,x=NULL,var.list=NULL,
       famdep=TRUE)
```

---

init.norm *computes initial values for the EM algorithm in the case of continuous measurements*

Description

computes initial values of means and variance-covariance matrices for the EM algorithm in the case of continuous measurements and multinormal model.

Usage

```
init.norm(y, K, x = NULL, var.list = NULL)
```

Arguments

`y`	a `n` times `d` matrix of continuous measurements, where `n` is the number of individuals and `d` is the number of measurements. All entries must be finite, if not an error is produced,
`K`	number of latent classes of the model,
`x`	a matrix of covariates if any, default is `NULL` (no covariates),
`var.list`	a list of integers indicating which covariates (taken from `x`) are used for a given measurement (a column of `y`).

Details

The function allocates every individual to a class by a simple clustering of the data and evaluates the means and variance-covariance matrices of measurements in each class. Treatment of covariates is not yet implemented, and any provided covariate value will be ignored.

Value

The function returns a list of 2 elements `mu` and `sigma` of length `K` each, `mu[k]` is the means vector (of length `d`) of measurements in class `k` and `sigma[k]` is the variances-covariances matrix (of dimension `d` times `d`) of measurements in class `k`.

Examples

```
#data
data(ped.cont)
status <- ped.cont[,6]
y <- ped.cont[status==2,7:ncol(ped.cont)]
#the function
init.norm(y,K=3)
```

`init.ordi` *computes the initial values for EM algorithm in the case of ordinal measurements*

Description

computes the initial values of cumulative logistic coefficients alpha for the EM algorithm in the case of ordinal measurements and a product multinomial model.

Usage

```
init.ordi(y, K, x = NULL, var.list = NULL)
```

Arguments

`y`	a `n` times `d` matrix of ordinal (or discrete) measurements, where `n` is the number of individuals and `d` is the number of measurements. All entries must be finite, if not an error is produced,
`K`	number of latent classes of the model,
`x`	a matrix of covariates if any, default is `NULL` (no covariates),
`var.list`	list of integers indicating which covariates (taken from `x`) are used for a given measurement (a column of `y`).

Details

The function allocates every individual to a class and evaluates the cumulative logistic coefficients for each measurement and each class. Regression coefficients for the covariates are set to 0.

Value

The function returns a list of one element `alpha` which is a list of `d` elements, each element `alpha[[j]]` is a `K` times `S-1` matrix, where `S` is the number of values of the measurement `y[,j]`, a row `alpha[[j]][k,]` gives the the cumulative logistic coefficients of class `k` and measurement `j` using `alpha.compute`.

See Also

alpha.compute

Examples

```
#data
data(ped.ordi)
status <- ped.ordi[,6]
y <- ped.ordi[,7:ncol(ped.ordi)]
#the function
init.ordi(y[status==2,],K=3)
```

`init.p.trans` *initializes the transition probabilities*

Description

initializes the marginal transition probabilities with or without parental constraint.

Usage

```
init.p.trans(K, trans.const = TRUE)
```

Arguments

`K`	number of latent classes,
`trans.const`	a logical variable indicating if the parental constraint is used. Parental constraint means that the class of a subject can be only one of his parents classes. Default is `TRUE`.

Details

All non-zero transition probabilities are set to be equal. The parental constraint indicator determines which transition probabilities are non-zero.

Value

the function returns `p.trans` an array of dimension `K` times `K+1` times `K+1`: `p.trans[c_i,c_1,c_2]` is the probability that the subject `i` is assigned to class `c_i` and his parents to classes `c_1` and `c_2`.

Examples

```
init.p.trans(3) #parental constraint is TRUE,
init.p.trans(3,trans.const=FALSE) #parental constraint is FALSE.
```

`lca.model` *fits latent class models for phenotypic measurements in pedigrees with or without familial dependence using an Expectation-Maximization (EM) algorithm*

Description

This is the main function for fitting latent class models. It performs some checks of the pedigrees (it exits if an individual has only one parent in the pedigree, if no children is in the pedigree or if there are not enough individuals for parameters estimation) and of the initial values (positivity of probabilites and their summation to one). For models with familial dependence, the child latent class depends on his parents classes via triplet-transition probabilities. In the case of models without familial dependence, it performs the classical Latent Class Analysis (LCA) where all individuals are supposed independent and the pedigree structure is meaningless. The EM algorithm stops when the difference between log-likelihood is smaller then `tol` that is fixed by the user.

Usage

```
lca.model(ped, probs, param, optim.param, fit = TRUE,
optim.probs.indic = c(TRUE, TRUE, TRUE, TRUE), tol = 0.001,
x = NULL, var.list = NULL, famdep = TRUE, modify.init = NULL)
```

Arguments

`ped`	a matrix or data frame representing pedigrees and measurements: `ped[,1]` family ID, `ped[,2]` subjects ID, `ped[,3]` dad ID, `ped[,4]` mom ID, `ped[,5]` sex, `ped[,6]` symptom status (2: symptomatic, 1: without symptoms, 0: missing), `ped[,7:ncol(ped)]` measurements, each column corresponds to a phenotypic measurement. If the measurement distribution specified with `optim.param` is multinomial, then these columns must either be of type `integer` of `factor`,
`probs`	a list of initial probability parameters (see below for more details). The function `init.p.trans` can be used to compute an initial value of the component `p.trans` of `probs`,
`param`	a list of initial measurement distribution parameters (see below for more details). The function `init.ordi` can be used to compute an initial value of `param` in the case of discrete or ordinal data (product multinomial distribution) and `init.norm` in the case of continous data (mutivariate normal distribution),
`optim.param`	a variable indicating how measurement distribution parameter optimization is performed (see below for more details),
`fit`	a logical variable, if `TRUE`, the EM algorithm is performed, if `FALSE`, only computation of weights and log-likelihood are performed with the initial parameter values without log-likelihood maximization,
`optim.probs.indic`	a vector of logical values indicating which probability parameters to estimate,
`tol`	a small number governing the stopping rule of the EM algorithm. Default is 0.001,

x	a matrix of covariates (optional), default is NULL,
var.list	a list of integers indicating the columns of x containing the covariates to use for a given phenotypic measurement, default is NULL,
famdep	a logical variable indicating if familial dependence model is used or not. Default is TRUE. In models without familial dependence, individuals are treated as independent and pedigree structure is meaningless. In models with familial dependence, a child class depends in his parents classes via a triplet-transition probability,
modify.init	a function to modify initial values of the EM algorithm, or NULL, default is NULL.

Details

The symptom status vector (column 6 of ped) takes value 1 for subjects that have been examined and show no symptoms (i.e. completely unaffected subjects). When applying the LCA to measurements available on all subjects, the status vector must take the value of 2 for every individual with measurements.

probs is a list of initial probability parameters:

For models with familial dependence:

p a probability vector, each p[c] is the probability that an symptomatic founder is in class c for c>=1,

p0 the probability that a founder without symptoms is in class 0,

p.trans an array of dimension K times K+1 times K+1, where K is the number of latent classes of the model, and is such that p.trans[c_i,c_1,c_2] is the conditional probability that a symptomatic individual i is in class c_i given that his parents are in classes c_1 and c_2,

p0connect a vector of length K, where p0connect[c] is the probability that a connector without symptoms is in class 0, given that one of his parents is in class c>=1 and the other in class 0,

p.found the probability that a founder is symptomatic,

p.child the probability that a child is symptomatic,

For models without familial dependence, all individuals are independent:

p a probability vector, each p[c] is the probability that an symptomatic individual is in class c for c>=1,

p0 the probability that an individual without symptoms is in class 0,

p.aff the probability that an individual is symptomatic,

param is a list of measurement distribution parameters: the coefficients alpha (cumulative logistic coefficients see alpha.compute) in the case of discrete or ordinal data, and means mu and variances-covariances matrices sigma in the case of continuous data,

optim.param is a variable indicating how the measurement distribution parameter estimation of the M step is performed. Two possibilities, optim.noconst.ordi and optim.const.ordi, are now available in the case of discrete or ordinal measurements, and four possibilities optim.indep.norm (measurements are independent, diagonal variance-covariance matrix), optim.diff.norm (general variance-covariance matrix but equal for all classes), optim.equal.norm (variance-covariance matrices are different for each class but equal variance and equal covariance for a class) and

optim.gene.norm (general variance-covariance matrices for all classes), are now available in the case of continuous measurements, One of the allowed values of optim.param must be entered without quotes.

optim.probs.indic is a vector of logical values of length 4 for models with familial dependence and 2 for models without familial dependence.

For models with familial dependence:

optim.probs.indic[1] indicates whether p0 will be estimated or not,

optim.probs.indic[2] indicates whether p0connect will be estimated or not,

optim.probs.indic[3] indicates whether p.found will be estimated or not,

optim.probs.indic[4] indicates whether p.connect will be estimated or not.

For models without familial dependence:

optim.probs.indic[1] indicates whether p0 will be estimated or not,

optim.probs.indic[2] indicates whether p.aff will be estimated or not.

All defaults are TRUE. If the dataset contains only nuclear families, there is no information to estimate p0connect and p.connect, and these parameters will not be estimated, irrespective of the indicator value.

Value

The function returns a list of 4 elements:

param	the Maximum Likelihood Estimator (MLE) of the measurement distribution parameters if fit=TRUE or the input param if fit=FALSE,
probs	the MLE of probability parameters if fit=TRUE or the input probs if fit=FALSE,

When measurements are available on all subjects, the probability parameters p0 and p0connect are degenerated to 0 and p.afound, p.child and p.aff to 1 in the output.

weight	an array of dimension n (the number of individuals) times 2 times K+1 (K being the number of latent classes in the selected model and the K+1th class being the unaffected class) giving the individual posterior probabilities. weight[i,s,c] is the posterior probability that individual i belongs to class c when his symptom status is s, where s takes two values: 1 for symptomatic and 2 for without symptom. In particular, all weight[,2,] are 0 for symptomatic individuals and all weight[,1,] are 0 for individuals without symptoms. For missing individuals (unkown symptom status), both weight[,1,] and weight[,2,] may be greater than 0.
ll	the maximum log-likelihood value (log-ML) if fit=TRUE or the log-likelihood computed with the input values of param and probs if fit=FALSE,

References

TAYEB, A. LABBE, A., BUREAU, A. and MERETTE, C. (2011) Solving Genetic Heterogeneity in Extended Families by Identifying Sub-types of Complex Diseases. Computational Statistics, 26(3): 539-560. DOI: 10.1007/s00180-010-0224-2,

LABBE, A., BUREAU, A. et MERETTE, C. (2009) Integration of Genetic Familial Dependence Structure in Latent Class Models. The International Journal of Biostatistics, 5(1): Article 6.

Examples

```
#data
data(ped.ordi)
fam <- ped.ordi[,1]
#probs and param
data(param.ordi)
data(probs)
#the function applied only to two first families of ped.ordi
lca.model(ped.ordi[fam%in%1:2,],probs,param.ordi,optim.noconst.ordi,
          fit=TRUE,optim.probs.indic=c(TRUE,TRUE,TRUE,TRUE),tol=0.001,x=NULL,
          var.list=NULL,famdep=TRUE,modify.init=NULL)
```

`model.select` *selects a latent class model for pedigree data*

Description

Performs selection of a latent class model for phenotypic measurements in pedigrees based on one of two possible methods: likelihood-based cross-validation or Bayesian Information Criterion (BIC) selection. This is the top-level function to perform a Latent Class Analysis (LCA), which calls the model fitting function `lca.model`. Model selection is performed among models within one of two types: with and without familial dependence. Two families of distributions are currently implemented: product multinomial for discrete (or ordinal) data and mutivariate normal for continuous data.

Usage

```
model.select(ped, distribution, trans.const = TRUE, optim.param,
             optim.probs.indic = c(TRUE, TRUE, TRUE, TRUE),
             famdep = TRUE, selec = "bic", H = 5, K.vec = 1:7,
             tol = 0.001, x = NULL, var.list = NULL)
```

Arguments

`ped`	a matrix containing variables coding the pedigree structure and the phenotype measurements: `ped[,1]` family ID, `ped[,2]` subjects ID, `ped[,3]` dad ID, `ped[,4]` mom ID, `ped[,5]` sex, `ped[,6]` symptom status (2: symptomatic, 1:

without symptoms, 0: missing), `ped[,7:ncol(ped)]` measurements, each column corresponds to a phenotypic measurement. If the argument `distribution` is `"multinomial"`, then these columns must either be of type `integer` of `factor`,

`distribution` a character variable taking the value `"normal"` for multivariate normal measurements and `"multinomial"` for ordinal or discrete multinomial measurements,

`trans.const` a logical variable indicating if the parental constraint is used. Parental constraint means that the class of a subject must be one of his parents classes. Default is `TRUE`,

`optim.param` a variable indicating how the measurement distribution parameter optimization is performed (see below for more details),

`optim.probs.indic` a vector of logical values indicating which probability parameters to estimate (see below for more details),

`famdep` a logical variable indicating if the familial dependence model is used or not. Default is `TRUE`. In models without familial dependence, individuals are treated as independent and pedigree structure is meaningless. In models with familial dependence, a child class depends in his parents classes via a triplet transition probability,

`selec` a character variables taking the value `bic` if BIC selection is used and the value `cross` if cross-validation is used,

`H` an integer giving the number of equal parts into which data will be splitted for the likelihood-based cross-validation model selection (see below for more details),

`K.vec` a vector of integers, the number of latent classes of candidate models, if `K.vec` has one value, only models with that number of classes will be fitted,

`tol` a small number governing the stopping rule of the EM algorithm. Default is 0.001,

`x` a matrix of covariates (optional), default is `NULL`,

`var.list` a list of integers indicating the columns of `x` containing the covariates to use for a given phenotypic measurement, default is `NULL`.

Details

In the case of cross-validation based-likelihood method, data is splitted into `H` parts: `H-1` parts as a training set and one part as a test set. For each model, a validation log-likelihood is obtained by evaluating the log-likelihood of the test set data using the parameter values estimated in the training set. This is repeated `H` times using a different part as training set each time, and a total validation log-likelihood is obtained by summation over the `H` test sets. The best model is the one having the largest validation log-likelihood. In the case of BIC selection method, the BIC is computed for each candidate model. The model with the smallest BIC is selected.

The symptom status vector (column 6 of `ped`) takes value 1 for subjects that have been examined and show no symptoms (i.e. completely unaffected subjects). When applying the LCA to measurements available on all subjects, the status vector must take the value of 2 for every individual with measurements. If covariates are used, covariate values must be provided for subjects with symptom status 0 (missing) but not for subjects with symptom status 1 (if covariate values are provided, they will be ignored).

optim.param is a variable indicating how the measurement distribution parameter optimization of the M step is performed. Two possibilities, optim.noconst.ordi and optim.const.ordi, are now available in the case of discrete or ordinal measurements, and four possibilities, optim.indep.norm (measurements are independent, diagonal variance-covariance matrix), optim.diff.norm (general variance-covariance matrix but equal for all classes), optim.equal.norm (variance-covariance matrices are different for each class but equal variance and equal covariance for a class) and optim.gene.norm (general variance-covariance matrices for all classes), in the case of continuous measurements. One of the allowed values of optim.param must be entered without quotes.

optim.probs.indic is a vector of logical values of length 4 for models with familial dependence and 2 for models without familial dependence indicating which probability parameters to estimate. See the help page for lca.model for a definition of the parameters.

For models with familial dependence:

optim.probs.indic[1] indicates whether p0 will be estimated or not,

optim.probs.indic[2] indicates whether p0connect will be estimated or not,

optim.probs.indic[3] indicates whether p.found will be estimated or not,

optim.probs.indic[4] indicates whether p.connect will be estimated or not.

For models without familial dependence:

optim.probs.indic[1] indicates whether p0 will be estimated or not,

optim.probs.indic[2] indicates whether p.aff will be estimated or not.

All defaults are TRUE.

Value

The function returns a list of 5 elements, the first 3 elements are common for BIC and cross-validation model selection methods and are:

param	the Maximum Likelihood Estimator (MLE) of the measurement distribution parameters of the selected model,
probs	the Maximum Likelihood Estimator (MLE) of the probability parameters of the selected model,
weight	an array of dimension n (the number of individuals) times 2 times K+1 (K being the number of latent classes in the selected model and the K+1th class being the unaffected class) giving the individual posterior probabilities. weight[i,s,c] is the posterior probability that individual i belongs to class c when his affection status is s, where s takes two values: 1 for symptomatic and 2 for without symptom. In particular, all weight[,2,] are 0 for symptomatic individuals and all weight[,1,] are 0 for individuals without symptoms. For missing individuals (unkown symptom status), both weight[,1,] and weight[,2,] may be greater than 0.

If the cross-validation selection method is used, the function returns also

ll	the value of the maximum log-likelihood (log-ML) of the selected model,
ll.valid	the total cross-validation log-likelihood of all candidate models,

and if the Bayesian Information Criterion selection method is used, the function returns also

`ll`	the value of maximum log-likelihood (log-ML) of all candidate models,
`bic`	the Bayesian Information Criterion `BIC=-2*log(ll)+m*log(n)` of all candidate models, where `m` is the number of free parameters of the model and `n` the total number of individuals.

References

TAYEB, A. LABBE, A., BUREAU, A. and MERETTE, C. (2011) Solving Genetic Heterogeneity in Extended Families by Identifying Sub-types of Complex Diseases. Computational Statistics, 26(3): 539-560. DOI: 10.1007/s00180-010-0224-2,

LABBE, A., BUREAU, A. et MERETTE, C. (2009) Integration of Genetic Familial Dependence Structure in Latent Class Models. The International Journal of Biostatistics, 5(1): Article 6.

See Also

See also `lca.model`.

Examples

```
#data
data(ped.cont)
fam <- ped.cont[,1]
#the function applied for the two first families of ped.cont
model.select(ped.cont[fam%in%1:2,],distribution="normal",trans.const=TRUE,
             optim.indep.norm,optim.probs.indic=c(TRUE,TRUE,TRUE,TRUE),
             famdep=TRUE,selec="bic",K.vec=1:3,tol=0.001,x=NULL,var.list=NULL)
```

---

n.param *computes the number of parameters of a model*

Description

computes the number of free parameters of a model, depending in the number of classes, the type of parameter optimization and the used of familial dependence, to be used in BIC model selection. This is an internal function not meant to be called by the user.

Usage

```
n.param(y, K, trans.const = TRUE, optim.param,
        optim.probs.indic = c(TRUE, TRUE, TRUE, TRUE), famdep = TRUE)
```

Arguments

y	a matrix of measurements,
K	an integer, the number of latent classes of a candiate model,
trans.const	a logical variable indicating if the parental constraint is used. Parental constraint means that the class of a subject can be only one of his parents classes. Default is TRUE,
optim.param	a function used for parameter optimization, see lca.model for more details,
optim.probs.indic	a vector of logical values indicating which probability parameters to be updated, see lca.model for more details,
famdep	a logical variable indicating if familial dependence model is used or not. Default is TRUE.

Value

The function returns the number of free parameters (of the measurement distribution and the probabilities of the latent classes).

See Also

See also model.select.

Examples

```
data(ped.cont)
y <- ped.cont[,7:ncol(ped.cont)]
n.param(y,K=3,trans.const=TRUE,optim.indep.norm,
        optim.probs.indic=c(TRUE,TRUE,TRUE,TRUE),famdep=TRUE)
```

---

optim.const.ordi *performs the M step for the measurement distribution parameters in multinomial case, with an ordinal constraint on the parameters*

Description

Estimates the cumulative logistic coefficients alpha in the case of multinomial (or ordinal) data with an ordinal constraint on the parameters.

Usage

```
optim.const.ordi(y, status, weight, param, x = NULL, var.list = NULL)
```

Arguments

y	a matrix of discrete (or ordinal) measurements (only for symptomatic subjects),
status	symptom status of all individuals,
weight	a matrix of n times K of individual weights, where n is the number of individuals and K is the total number of latent classes in the model,
param	a list of measurement density parameters, here is a list of alpha,
x	a matrix of covariates (optional). Default id NULL,
var.list	a list of integers indicating which covariates (taken from x) are used for a given type of measurement

Details

the constraint on the parameters is that, for a symptom j, the rows alpha[[j]][k,] are equal for all classes k except the first values. Therefore, maximum likelihood estimators are not explicit and the function lrm of the package rms is used to perform a numerical optimization.

Value

The function returns a list of estimated parameters param satisfying the constraint.

Examples

```
#data
data(ped.ordi)
status <- ped.ordi[,6]
y <- ped.ordi[,7:ncol(ped.ordi)]
data(peel)
#probs and param
data(probs)
data(param.ordi)
#e step
weight <- e.step(ped.ordi,probs,param.ordi,dens.prod.ordi,peel,x=NULL,
                 var.list=NULL,famdep=TRUE)$w
weight <- matrix(weight[,1,1:length(probs$p)],nrow=nrow(ped.ordi),
                 ncol=length(probs$p))
#the function
optim.const.ordi(y[status==2,],status,weight,param.ordi,x=NULL,
                 var.list=NULL)
```

---

optim.diff.norm *performs the M step for measurement density parameters in multinormal case*

Description

Estimates the mean mu and parameters of the variance-covariance matrix sigma of a multinormal distribution for the measurements with a general variance-covariance matrix identical for all classes.

Usage

```
optim.diff.norm(y, status, weight, param, x = NULL, var.list = NULL)
```

Arguments

y	a matrix of continuous measurements (only for symptomatic subjects),
status	symptom status of all individuals,
weight	a matrix of n times K of individual weights, where n is the number of individuals and K is the total number of latent classes in the model,
param	a list of measurement density parameters, here is a list of mu and sigma,
x	a matrix of covariates (optional). Default id NULL,
var.list	a list of integers indicating which covariates (taken from x) are used for a given type of measurement.

Details

The values of explicit estimators are computed for both mu and sigma. The variance-covariance matrices sigma are identical for all classes. Treatment of covariates is not yet implemented, and any provided covariate value will be ignored.

Value

The function returns a list of estimated parameters param.

Examples

```
#data
data(ped.cont)
status <- ped.cont[,6]
y <- ped.cont[,7:ncol(ped.cont)]
data(peel)
#probs and param
data(probs)
data(param.cont)
#e step
weight <- e.step(ped.cont,probs,param.cont,dens.norm,peel,x=NULL,
                 var.list=NULL,famdep=TRUE)$w
weight <- matrix(weight[,1,1:length(probs$p)],nrow=nrow(ped.cont),
                 ncol=length(probs$p))
#the function
optim.diff.norm(y[status==2,],status,weight,param.cont,x=NULL,
                 var.list=NULL)
```

`optim.equal.norm` *performs the M step for measurement density parameters in multinormal case*

Description

Estimates the mean `mu` and parameters of the variance-covariance matrix `sigma` of a multinormal distribution for the measurements with equal variance for all measurements and equal covariance between all pairs of measurements within each class. The variance and covariance parameters are however distinct for each class.

Usage

```
optim.equal.norm(y, status, weight, param, x = NULL, var.list = NULL)
```

Arguments

`y`	a matrix of continuous measurements (only for symptomatic subjects),
`status`	symptom status of all individuals,
`weight`	a matrix of `n` times `K` of individual weights, where `n` is the number of individuals and `K` is the total number of latent classes in the model,
`param`	a list of measurement density parameters, here is a list of `mu` and `sigma`,
`x`	a matrix of covariates (optional). Default id `NULL`,
`var.list`	a list of integers indicating which covariates (taken from `x`) are used for a given type of measurement.

Details

The values of explicit estimators are computed for both `mu` and `sigma`. The variance-covariance matrices `sigma` are distinct for each class. Treatment of covariates is not yet implemented, and any provided covariate value will be ignored.

Value

The function returns a list of estimated parameters `param`.

Examples

```
#data
data(ped.cont)
status <- ped.cont[,6]
y <- ped.cont[,7:ncol(ped.cont)]
data(peel)
#probs and param
data(probs)
data(param.cont)
#e step
```

```
weight <- e.step(ped.cont,probs,param.cont,dens.norm,peel,x=NULL,
                 var.list=NULL,famdep=TRUE)$w
weight <- matrix(weight[,1,1:length(probs$p)],nrow=nrow(ped.cont),
                 ncol=length(probs$p))
#the function
optim.equal.norm(y[status==2,],status,weight,param.cont,x=NULL,
                 var.list=NULL)
```

`optim.gene.norm` *performs the M step for measurement density parameters in multinormal case*

Description

Estimates the mean `mu` and parameters of the variance-covariance matrix `sigma` of a multinormal distribution for the measurements with general variance-covariance matrices distinct for each class.

Usage

```
optim.gene.norm(y, status, weight, param, x = NULL, var.list = NULL)
```

Arguments

`y`	a matrix of continuous measurements (only for symptomatic subjects),
`status`	symptom status of all individuals,
`weight`	a matrix of `n` times `K` of individual weights, where `n` is the number of individuals and `K` is the total number of latent classes in the model,
`param`	a list of measurement density parameters, here is a list of `mu` and `sigma`,
`x`	a matrix of covariates (optional). Default id `NULL`,
`var.list`	a list of integers indicating which covariates (taken from `x`) are used for a given type of measurement.

Details

The values of explicit estimators are computed for both `mu` and `sigma`. This is the general case, the variance-covariance matrices `sigma` of the different classes are distinct and unconstrained. Treatment of covariates is not yet implemented, and any provided covariate value will be ignored.

Value

The function returns a list of estimated parameters `param`.

Examples

```
#data
data(ped.cont)
status <- ped.cont[,6]
y <- ped.cont[,7:ncol(ped.cont)]
data(peel)
#probs and param
data(probs)
data(param.cont)
#e step
weight <- e.step(ped.cont,probs,param.cont,dens.norm,peel,x=NULL,
                 var.list=NULL,famdep=TRUE)$w
weight <- matrix(weight[,1,1:length(probs$p)],nrow=nrow(ped.cont),
                 ncol=length(probs$p))
#the function
optim.gene.norm(y[status==2,],status,weight,param.cont,x=NULL,
                 var.list=NULL)
```

`optim.indep.norm` *performs the M step for measurement density parameters in multinormal case*

Description

Estimates the mean `mu` and parameters of the variance-covariance matrix `sigma` of a multinormal distribution for the measurements with diagonal variance-covariance matrices for each class, i.e. measurements are supposed independent.

Usage

```
optim.indep.norm(y, status, weight, param, x = NULL, var.list = NULL)
```

Arguments

`y`	a matrix of continuous measurements (only for symptomatic subjects),
`status`	symptom status of all individuals,
`weight`	a matrix of `n` times `K` of individual weights, where `n` is the number of individuals and `K` is the total number of latent classes in the model,
`param`	a list of measurement density parameters, here is a list of `mu` and `sigma`,
`x`	a matrix of covariates (optional). Default id `NULL`,
`var.list`	a list of integers indicating which covariates (taken from `x`) are used for a given type of measurement.

Details

The values of explicit estimators are computed for both `mu` and `sigma`. All variance-covariance matrices `sigma` are diagonal, i.e. measurements are supposed independent. Treatment of covariates is not yet implemented, and any provided covariate value will be ignored.

Value

The function returns a list of estimated parameters `param`.

Examples

```
#data
data(ped.cont)
status <- ped.cont[,6]
y <- ped.cont[,7:ncol(ped.cont)]
data(peel)
#probs and param
data(probs)
data(param.cont)
#e step
weight <- e.step(ped.cont,probs,param.cont,dens.norm,peel,x=NULL,
                 var.list=NULL,famdep=TRUE)$w
weight <- matrix(weight[,1,1:length(probs$p)],nrow=nrow(ped.cont),
                 ncol=length(probs$p))
#the function
optim.indep.norm(y[status==2,],status,weight,param.cont,x=NULL,
                 var.list=NULL)
```

`optim.noconst.ordi` *performs the M step for the measurement distribution parameters in multinomial case without constraint on the parameters*

Description

Estimates the cumulative logistic coefficients `alpha` in the case of multinomial (or ordinal) data without constraint on the coefficients.

Usage

```
optim.noconst.ordi(y, status, weight, param, x = NULL, var.list = NULL)
```

Arguments

`y`	a matrix of discrete (or ordinal) measurements (only for symptomatic subjects),
`status`	symptom status of all individuals,
`weight`	a matrix of `n` times `K` of individual weights, where `n` is the number of individuals and `K` is the total number of latent classes in the model,
`param`	a list of measurement distribution parameters, here is a list `alpha` of cumulative logistic coefficients,
`x`	a matrix of covariates (optional). Default is `NULL`,
`var.list`	a list of integers indicating which covariates (taken from `x`) are used for a given type of measurment.

Details

The values of explicit estimators are computed by logistic transformation of weighted empirical frequencies.

Value

the function returns a list of estimated parameters `param`.

Examples

```
#data
data(ped.ordi)
status <- ped.ordi[,6]
y <- ped.ordi[,7:ncol(ped.ordi)]
data(peel)
#probs and param
data(probs)
data(param.ordi)
#e step
weight <- e.step(ped.ordi,probs,param.ordi,dens.prod.ordi,peel,x=NULL,
                 var.list=NULL,famdep=TRUE)$w
weight <- matrix(weight[,1,1:length(probs$p)],nrow=nrow(ped.ordi),
                 ncol=length(probs$p))
#the function
optim.noconst.ordi(y[status==2,],status,weight,param.ordi,x=NULL,
                   var.list=NULL)
```

`optim.probs` *performs the M step of the EM algorithm for the probability parameters*

Description

estimates the probability parameters (`p`, `p.trans`, `p0`,...) in the M step of the EM algorithm in both cases with and without familial dependence. This is an internal function not meant to be called by the user.

Usage

```
optim.probs(ped, probs, optim.probs.indic = c(TRUE, TRUE, TRUE, TRUE),
            res.weight, famdep = TRUE)
```

Arguments

`ped`	a matrix of pedigrees data, see `e.step` for more details,
`probs`	all probability parameters to be optimized,
`optim.probs.indic`	a vector of logical values indicating which probability parameters to be updated,

res.weight	a matrix of n times K, individual weights, where n is the number of individuals and K is the total number of latent classes in the model, resulting from the E step of the EM algorithm (see e.step)
famdep	a logical variable indicating if familial dependence model is used or not. Default is TRUE.

Details

explicit estimators are computed in function of the weights.

Value

the function returns the estimated probs of all probability parameters.

References

TAYEB et al.: Solving Genetic Heterogeneity in Extended Families by Identifying Sub-types of Complex Diseases. Computational Statistics, 2011, DOI: 10.1007/s00180-010-0224-2.

Examples

```
#data
data(ped.cont)
data(peel)
#probs and param
data(probs)
data(param.cont)
#e step
weight <- e.step(ped.cont,probs,param.cont,dens.norm,peel,x=NULL,
                 var.list=NULL,famdep=TRUE)
#the function
optim.probs(ped.cont,probs,weight,optim.probs.indic=
            c(TRUE,TRUE,TRUE,TRUE),famdep=TRUE)
```

---

p.compute *computes the probability vector using logistic coefficients*

Description

computes the probability vector using cumulative logistic coefficients

Usage

```
p.compute(alpha,decal)
```

Arguments

alpha	a vector of cumulative logistic coefficients, the first value can be -Inf, followed by, eventually, only one negative value, then only positive values. It can end by Inf values.
decal	offset term to be applied to sums of logistic coefficients

Details

If alpha has S-1 values, p.compute returns p of length S. If Y is a random variable taking values in {1,...,S} with probabilities p, coefficients alpha[i] are given by:

$$p_1 + ... + p_i = P(Y \leq i) = \frac{\exp(\alpha_1 + ... + \alpha_i)}{(1 + \exp(\alpha_1 + ... + \alpha_i)}$$

for all i<=S-1.

Value

p: a probability vector

See Also

p.compute is the inverse function of alpha.compute

Examples

```
# a vector of probability
p <- c(0,0.2,0,0,0.3,0.4,0.1,0,0)
alpha <- alpha.compute(p)
#gives alpha= -Inf -1.38 0 0 1.38 0 2.19 Inf Inf
p.compute(alpha) #gives p
```

---

p.post.child *computes the posterior probability of observations of a child*

Description

computes the posterior probability of measurements of a child for each class and each symptom status of the subject given the classes of both of his parents. This is an internal function not meant to be called by the user.

Usage

```
p.post.child(child, c.connect, c.spouse, status, probs, fyc)
```

Arguments

child	a child in the pedigree,
c.connect	the class of one parent (who is a connector) of the child,
c.spouse	the class of the other parent of the child,
status	the symptom status vector of the whole pedigree,
probs	a list of all probability parameters of the model,
fyc	a matrix of n times K+1 giving the density of measurements of each individual if allocated to class k, where n is the number of individuals and K is the total number of latent classes in the model,

Value

the function returns p.child a matrix of 2 times K+1 entries such that p.child[s,k] is the posterior probability of the measurements Y_child under status S_child=s and when he is assigned to class k and his parents are assigned to classes c.connect and c.spouse.

References

TAYEB et al.: Solving Genetic Heterogeneity in Extended Families by Identifying Sub-types of Complex Diseases. Computational Statistics, 2011, DOI: 10.1007/s00180-010-0224-2.

Examples

```
#data
data(ped.cont)
fam <- ped.cont[,1]
dad <- ped.cont[fam==1,3]
status <- ped.cont[fam==1,6]
y <- ped.cont[fam==1,7:ncol(ped.cont)]
#a child
child <- which(dad!=0)[1]
data(probs)
data(param.cont)
#densities of the observations
fyc <- matrix(1,nrow=nrow(y),ncol=length(probs$p)+1)
fyc[status==2,1:length(probs$p)] <- t(apply(y[status==2,],1,dens.norm,
                                            param.cont,NULL))
#the function
p.post.child(child,c.connect=1,c.spouse=3,status,probs,fyc)
```

---

p.post.found *computes the posterior probability of observations of a founder*

Description

computes the posterior probability of measurements of a founder for each class and each symptom status of the founder. This is an internal function not meant to be called by the user.

Usage

```
p.post.found(found, status, probs, fyc)
```

Arguments

found	a founder in the pedigree (individual without parents in the pedigree),
status	the symptom status vector of the whole pedigree,
probs	a list of all probability parameters of the model,
fyc	a matrix of n times K+1 given the density of measurements of each individual if allocated to class k, where n is the number of individuals and K is the total number of latent classes in the model,

Value

the function returns p.found a matrix of 2 times K+1 entries: p.found[s,k] is the posterior probability of the observations Y_found under status S_found=s and when he is assigned to class k.

References

TAYEB et al.: Solving Genetic Heterogeneity in Extended Families by Identifying Sub-types of Complex Diseases. Computational Statistics, 2001, DOI: 10.1007/s00180-010-0224-2.

Examples

```
#data
data(ped.cont)
fam <- ped.cont[,1]
dad <- ped.cont[fam==1,3]
status <- ped.cont[fam==1,6]
y <- ped.cont[fam==1,7:ncol(ped.cont)]
#a founder
found <- which(dad==0)[1]
data(probs)
data(param.cont)
#densities of the observations
fyc <- matrix(1,nrow=nrow(y),ncol=length(probs$p)+1)
fyc[status==2,1:length(probs$p)] <- t(apply(y[status==2,],1,dens.norm,
                                            param.cont,NULL))
#the function
p.post.found(found,status,probs,fyc)
```

---

param.cont *parameters to be used for examples in the case of continuous measurements*

Description

means and variance-covariance matrices for each class to be used in examples with continuous measurements.

Usage

```
data(param.cont)
```

Details

`ped.param` is a list of 2 elements:

`mu` a list of `K=3` (the number of latent classes) entries, each represents the means of the measurement multinormal density in the class,

`sigma` a list of `K=3` entries, each is the variance-covariance matrix of the measurement multinormal density in the class.

The dimension (the number of multinormal measurements) used in the dataset is 4.

See Also

See also `init.norm`

Examples

```
data(param.cont)
```

---

param.ordi *parameters to be used for examples in the case of discrete or ordinal measurements*

Description

list of cumulative logistic coefficients for each measurement and each class to be used in examples for discrete or ordinal models.

Usage

```
data(param.ordi)
```

Details

`ped.param` is a list of 1 element: `alpha` a list of `d=4` (the number of measurements) entries, each is a matrix of `K=3` (the number of classes) times `S[j]` (the number of possible values of measurement `j`), a row `alpha[[j]][k,]` contains the logistic coefficients of the measurement `j` for class `k`.

See Also

See also `init.ordi`

Examples

```
data(param.ordi)
```

ped.cont *pedigrees with continuous data to be used for examples*

Description

data set of 48 pedigrees: a matrix of pedigrees data with continuous observations to be used for examples.

Usage

```
data(ped.cont)
```

Details

ped is a matrix of 10 columns: ped[,1] family ID, ped[,2] subject ID, ped[,3] father ID, 0 for founders (i.e. subjects having no parents in the pedigree), ped[,4] mother ID, 0 for founders (i.e. subjects having no parents in the pedigree), ped[,5] subject sex: 1 male, 2 female, ped[,6] symptom status (2: symptomatic, 1: without symptoms, 0: missing), ped[,7:10] continuous observations, NA for missing and without symptoms,

Examples

```
data(ped.cont)
```

ped.ordi *pedigrees with discrete or ordinal data to be used for examples*

Description

data set of 48 pedigrees: a matrix of pedigrees data with discrete or ordinal observations to be used for examples.

Usage

```
data(ped.ordi)
```

Details

ped is a matrix of 10 columns: ped[,1] family ID, ped[,2] subject ID, ped[,3] father ID, 0 for founders (i.e. subjects having no parents in the pedigree), ped[,4] mother ID, 0 for founders (i.e. subjects having no parents in the pedigree), ped[,5] subject sex: 1 male, 2 female, ped[,6] symptom status (2: symptomatic, 1: without symptoms, 0: missing), ped[,7:10] discrete or ordinal observations, NA for missing and without symptoms,

Examples

```
data(ped.ordi)
```

peel *peeling order of pedigrees and couples in pedigrees*

Description

peel is a list of 48 entries, each gives the peeling order of the pedigrees and lists the couples in the 48 pedigrees of ped.cont and peed.ordi.

Usage

```
data(peel)
```

Value

For a pedigree f in the data ped.cont or ped.ordi, peel[[f]] is a list of three entries:

generation the number of generations in the pedigree,

peel.connect a matrix of generation rows, each giving the connectors of the generation in the order of peeling,

couple a matrix of two columns, giving the couples in the pedigree.

See Also

See also ped.cont and ped.ordi.

Examples

```
data(peel)
```

probs *probabilities parameters to be used for examples*

Description

a list of probability parameters such as the probability that a founder is assigned to each class, the transition probabilities and the probability that a child is symptomatic.

Usage

```
data(probs)
```

Details

probs a list of probability parameters:

For models with familial dependence:

p a probability vector, each p[c] is the probability that an symptomatic founder is in class c for c>=1,

p0 the probability that a founder without symptoms is in class 0,

p.trans an array of dimension K times K+1 times K+1, where K is the number of latent classes of the model, and is such that p.trans[c_i,c_1,c_2] is the conditional probability that a symptomatic individual i is in class c_i given that his parents are in classes c_1 and c_2,

p0connect a vector of length K, where p0connect[c] is the probability that a connector without symptoms is in class 0, given that one of his parents is in class c>=1 and the other in class 0,

p.found the probability that a founder is symptomatic,

p.child the probability that a child is symptomatic,

For models without familial dependence, all individuals are independent:

p a probability vector, each p[c] is the probability that an symptomatic individual is in class c for c>=1,

p0 the probability that an individual without symptoms is in class 0,

p.aff the probability that an individual is symptomatic,

Examples

```
data(probs)
```

---

upward *performs the upward step of the peeling algorithm of a pedigree*

Description

computes the probability of observations below connectors conditionally to their classes given the model parameters. This is an internal function not meant to be called by the user.

Usage

```
upward(id, dad, mom, status, probs, fyc, peel)
```

Arguments

id	individual ID of the pedigree,
dad	dad ID,
mom	mom ID,
status	symptom status: (2: symptomatic, 1: without symptoms, 0: missing),
probs	a list of probability parameters of the model,
fyc	a matrix of n times K+1 given the density of observations of each individual if allocated to class k, where n is the number of individuals and K is the total number of latent classes in the model,
peel	a list of pedigree peeling result containing connectors by peeling order and couples of parents.

Details

This function computes the probability of observations below connectors conditionally to their classes using the function upward.connect

Value

The function returns a list of 2 elements:

sum.child	an array of dimension n times K+1 times K+1 such that sum.child[i,c_1,c_2] is the probability of individual i measurements when his parent are assigned to classes c_1 and c_2,
p.yF.c	an array of dimension n times 2 times K+1 giving the probability of all measurements below the individual, depending on his status and his class.

References

TAYEB et al.: Solving Genetic Heterogeneity in Extended Families by Identifying Sub-types of Complex Diseases. Computational Statistics, 2011, DOI: 10.1007/s00180-010-0224-2.

See Also

See also upward.connect

Examples

```
#data
data(ped.cont)
data(peel)
fam <- ped.cont[,1]
id <- ped.cont[fam==1,2]
dad <- ped.cont[fam==1,3]
mom <- ped.cont[fam==1,4]
status <- ped.cont[fam==1,6]
y <- ped.cont[fam==1,7:ncol(ped.cont)]
peel <- peel[[1]]
```

```
#standardize id to be 1, 2, 3, ...
id.origin <- id
standard <- function(vec) ifelse(vec%in%id.origin,which(id.origin==vec),0)
id <- apply(t(id),2,standard)
dad <- apply(t(dad),2,standard)
mom <- apply(t(mom),2,standard)
peel$couple <- cbind(apply(t(peel$couple[,1]),2,standard),
                     apply(t(peel$couple[,2]),2,standard))
for(generat in 1:peel$generation)
peel$peel.connect[generat,] <- apply(t(peel$peel.connect[generat,]),2,standard)
#probs and param
data(probs)
data(param.cont)
#densities of the observations
fyc <- matrix(1,nrow=length(id),ncol=length(probs$p)+1)
fyc[status==2,1:length(probs$p)] <- t(apply(y[status==2,],1,dens.norm,
                                      param.cont,NULL))
#the function
upward(id,dad,mom,status,probs,fyc,peel)
```

`upward.connect` *performs the upward step for a connector*

Description

computes the probability of the measurements below a connector conditionally to the connector latent class given the model parameters. This is an internal function not meant to be called by the user.

Usage

```
upward.connect(connect, spouse.connect, children.connect, status,
probs, p.yF.c, fyc, sum.child)
```

Arguments

`connect`	a connector in the pedigree,
`spouse.connect`	spouse of the connector,
`children.connect`	children of the connector,
`status`	a vector of symptom status of the whole pedigree,
`probs`	a list of probability parameters of the model,
`p.yF.c`	an array of dimension `n` times 2 times `K+1` giving the probability of measurements below the individual, depending on his status and his class, where `n` is the number of individuals and `K` is the total number of latent classes in the model,
`fyc`	a matrix of `n` times `K+1` given the density of measurements of each individual if allocated to class `k`,

sum.child — an array of dimension nber.indiv times K+1 times K+1 such that sum.child[i,c_1,c_2] is the probability of individual i measurements when his parent are assigned to classes c_1 and c_1.

Details

If Y_above(i) is the observations below connector i and C_i is his class, the functions computes P(Y_below(i)|C_i).

Value

The function returns a list of 2 elements:

sum.child — an array of dimension n times K+1 times K+1 such that sum.child[i,c_1,c_2] is the probability of individual i observations when his parent are assigned to classes c_1 and c_2,

p.yF.c — a array of dimension n times 2 times K+1 giving the probability of measurements below the individual, depending on his status and his class, updated for the current connector.

References

TAYEB et al.: Solving Genetic Heterogeneity in Extended Families by Identifying Sub-types of Complex Diseases. Computational Statistics, 2011, DOI: 10.1007/s00180-010-0224-2.

See Also

See also upward

Examples

```
#data
data(ped.cont)
data(peel)
fam <- ped.cont[,1]
id <- ped.cont[fam==1,2]
dad <- ped.cont[fam==1,3]
mom <- ped.cont[fam==1,4]
status <- ped.cont[fam==1,6]
y <- ped.cont[fam==1,7:ncol(ped.cont)]
peel <- peel[[1]]
#standardize id to be 1, 2, 3, ...
id.origin <- id
standard <- function(vec) ifelse(vec%in%id.origin,which(id.origin==vec),0)
id <- apply(t(id),2,standard)
dad <- apply(t(dad),2,standard)
mom <- apply(t(mom),2,standard)
peel$couple <- cbind(apply(t(peel$couple[,1]),2,standard),
                     apply(t(peel$couple[,2]),2,standard))
for(generat in 1:peel$generation)
peel$peel.connect[generat,] <- apply(t(peel$peel.connect[generat,]),2,standard)
```

```
#a nuclear family
#connector in the pedigree 1
connect <- peel$peel.connect[1,1]
#soupse of connector connect
spouse.connect <- peel$couple[peel$couple[,1]==connect,2]
#children of connector connect
children.connect <- union(id[dad==connect],id[mom==connect])
#probs and param
data(probs)
data(param.cont)
#probabilitiy of observations above
p.yF.c <- matrix(1,nrow=length(id),ncol=length(probs$p)+1)
#densities of the observations
fyc <- matrix(1,nrow=length(id),ncol=length(probs$p)+1)
fyc[status==2,1:length(probs$p)] <- t(apply(y[status==2,],1,dens.norm,
                                      param.cont,NULL))
#sums over childs
sum.child <- array(0,c(length(id),length(probs$p)+1,length(probs$p)+1))
#the function
upward.connect(connect,spouse.connect,children.connect,status,probs,
               p.yF.c,fyc,sum.child)
```

`weight.famdep` *performs the computation of triplet and individual weights for a pedigree under familial dependence*

Description

computes the triplet and the individual weights of the E step of the EM algorithm for a pedigree in the case of familial dependence. It returns also the overall log-likelihood of the observations. This is an internal function not meant to be called by the user.

Usage

```
weight.famdep(id, dad, mom, status, probs, fyc, peel)
```

Arguments

`id`	individual ID of the pedigree,
`dad`	dad ID,
`mom`	mom ID,
`status`	symptom status: (2: symptomatic, 1: without symptoms, 0: missing),
`probs`	list of probability parameters of the model,
`fyc`	a matrix of `n` times `K+1` given the density of observations of each individual if allocated to class `k`, where `n` is the number of individuals and `K` is the total number of latent classes in the model,
`peel`	a list of pedigree peeling containing connectors by peeling order and couples of parents

Details

the function calls the functions upward and downward which perform the required probability computations by processing the pedigree by nuclear family (or equivalently by connector) following the peeling order.

Value

the function returns a list of 3 elements:

ww	triplet weights: an array of n times 2 times K+1 times K+1 times K+1, see e.step,
w	individual weights: an array of n times 2 times K+1, see e.step,
ll	log-likelihood.

References

TAYEB et al.: Solving Genetic Heterogeneity in Extended Families by Identifying Sub-types of Complex Diseases. Computational Statistics, 2011, DOI: 10.1007/s00180-010-0224-2.

See Also

See also upward, downward, e.step.

Examples

```
#data
data(ped.cont)
data(peel)
fam <- ped.cont[,1]
id <- ped.cont[fam==1,2]
dad <- ped.cont[fam==1,3]
mom <- ped.cont[fam==1,4]
status <- ped.cont[fam==1,6]
y <- ped.cont[fam==1,7:ncol(ped.cont)]
peel <- peel[[1]]
#probs and param
data(probs)
data(param.cont)
#densities of the observations
fyc <- matrix(1,nrow=length(id),ncol=length(probs$p)+1)
fyc[status==2,1:length(probs$p)] <- t(apply(y[status==2,],1,dens.norm,
                                    param.cont,NULL))
#the function
weight.famdep(id,dad,mom,status,probs,fyc,peel)
```

`weight.nuc` *performs the computation of unnormalized triplet and individuals weights for a nuclear family in the pedigree*

Description

the weighting algorithm proceeds by nuclear family, the function `weight.nuc` computes the unnormalized triplet and individuals weights for a nuclear family in the pedigree. This is an internal function not meant to be called by the user.

Usage

```
weight.nuc(connect, spouse.connect, children.connect, status,
probs, fyc, p.ybarF.c, ww, w, res.upward)
```

Arguments

`connect`	a connector in the pedigree,
`spouse.connect`	spouse of the connector,
`children.connect`	children of the connector,
`status`	vector of symptom status of the whole pedigree,
`probs`	all probability parameters of the model,
`fyc`	a matrix of `n` times `K+1` given the density of observations of each individual if allocated to class `k`, where `n` is the number of individuals and `K` is the total number of latent classes in the model,
`p.ybarF.c`	a array of dimension `n` times 2 times `K+1` giving the probability of observations above the individual, depending on his status and his class and conditionally on his class,
`ww`	unnormalized triplet weights, an array of `n` times 2 times `K+1` times `K+1` times `K+1`, where `n` is the number of individuals and `K` is the total number of latent classes in the model, see `e.step`,
`w`	unnormalized individual weights, an array of `n` times 2 times `K+1`, see `e.step`,
`res.upward`	result of the upward step of the weighting algorithm, see `upward`,

Details

updated `ww` and `w` are computed for the current nuclear family.

Value

the function returns a list of 2 elements:

`ww`	unnormalized triplet weights, an array of `n` times 2 times `K+1` times `K+1` times `K+1`, see `e.step`,
`w`	unnormalized individual weights, an array of `n` times 2 times `K+1`, see `e.step`.

References

TAYEB et al.: Solving Genetic Heterogeneity in Extended Families by Identifying Sub-types of Complex Diseases. Computational Statistics, 2011, DOI: 10.1007/s00180-010-0224-2.

See Also

See also downward

Examples

```
#data
data(ped.cont)
data(peel)
fam <- ped.cont[,1]
id <- ped.cont[fam==1,2]
dad <- ped.cont[fam==1,3]
mom <- ped.cont[fam==1,4]
status <- ped.cont[fam==1,6]
y <- ped.cont[fam==1,7:ncol(ped.cont)]
peel <- peel[[1]]
#standardize id to be 1, 2, 3, ...
id.origin <- id
standard <- function(vec) ifelse(vec%in%id.origin,which(id.origin==vec),0)
id <- apply(t(id),2,standard)
dad <- apply(t(dad),2,standard)
mom <- apply(t(mom),2,standard)
peel$couple <- cbind(apply(t(peel$couple[,1]),2,standard),
                     apply(t(peel$couple[,2]),2,standard))
for(generat in 1:peel$generation)
peel$peel.connect[generat,] <- apply(t(peel$peel.connect[generat,]),2,standard)
#the first nuclear family
generat <- peel$generation
connect <- peel$peel.connect[generat,]
connect <- connect[connect>0]
spouse.connect <- peel$couple[peel$couple[,1]==connect,2]
children.connect <- union(id[dad==connect],id[mom==connect])
#probs and param
data(probs)
data(param.cont)
#densities of the observations
fyc <- matrix(1,nrow=length(id),ncol=length(probs$p)+1)
fyc[status==2,1:length(probs$p)] <- t(apply(y[status==2,],1,dens.norm,
                                            param.cont,NULL))
#triplet and individual weights
ww <- array(0,dim=c(length(id),rep(2,3),rep(length(probs$p)+1,3)))
w <- array(0,dim=c(length(id),2,length(probs$p)+1))
#probability of the observations below
p.ybarF.c <- array(1,dim=c(length(id),2,length(probs$p)+1))
p.ybarF.c[connect,,] <- p.post.found(connect,status,probs,fyc)
#the upward step
res.upward <- upward(id,dad,mom,status,probs,fyc,peel)
#the function
```

```
weight.nuc(connect,spouse.connect,children.connect,status,probs,fyc,
           p.ybarF.c,ww,w,res.upward)
```

Index

alpha.compute, 2, *11*, *13, 14*, *16*, *31*
attrib.dens, 3

dens.norm, 4
dens.prod.ordi, 5
downward, 6, *9*, *42*, *44*
downward.connect, *7*, 8

e.step, *7*, 10, *29, 30*, *42, 43*

init.norm, 12, *15*, *34*
init.ordi, *5, 6*, 13, *15*, *34*
init.p.trans, 14, *15*

lca.model, *12*, 15, *20–22*

model.select, 18, *22*

n.param, 21

optim.const.ordi, *4*, *16*, *20*, 22
optim.diff.norm, *4*, *16*, *20*, 23
optim.equal.norm, *4*, *16*, *20*, 25
optim.gene.norm, *4*, *17*, *20*, 26
optim.indep.norm, *4*, *16*, *20*, 27
optim.noconst.ordi, *4*, *16*, *20*, 28
optim.probs, 29

p.compute, *3*, 30
p.post.child, 31
p.post.found, 32
param.cont, 33
param.ordi, 34
ped.cont, 35, *36*
ped.ordi, 35, *36*
peel, 36
probs, 36

upward, *6*, *8*, 37, *40*, *42, 43*
upward.connect, *38*, 39

weight.famdep, *12*, 41
weight.nuc, *7*, 43