Package 'phenix'

April 14, 2016

Type Package

Title Missing Phenotype Imputation

Version 1.0

Date 2014-10-03

Author Andy Dahl

Maintainer Andy Dahl <andywdahl@gmail.com>

Description Estimates the missing entries in a matrix of phenotypes using genetic information summarized by a kinship matrix. Also estimates breeding values and trait heritabilities.

License file LICENSE

R topics documented:

kronsum 1
LMM 3
misc 5
MPMM 6
MPMM_helpers 8
MPMM_impute 9
MVN_impute 10
mv_gwas 12
phenix 14
phenix_vb 16
sim_data 17

Index **19**

`kronsum` *Kronecker Sum Operations with Low Complexity*

Description

Efficiently perform partial trace and matrix-vector multiplications with a (possibly inverse) Kronecker sum of symmetric matrices

Usage

```
KS_dot_vec( A, B, C, inv = FALSE, svd.A = svd(A), svd.C = svd(C), diag.A=FALSE )
trp_KS( keep, kill, svd.keep = svd(keep), lam.kill = svd(kill)$d )
tr(X)
```

Arguments

A	The N x N, symmetric left Kronecker summand
C	The P x P, symmetric right Kronecker summand
B	An N x P matrix
X	A square matrix
inv	Whether the Kronecker sum is inverted
svd.A	Optional svd of A
svd.C	Optional svd of C
diag.A	Whether A is diagonal
keep	The Kronecker summand whose dimension is returned by the partial trace
kill	The Kronecker summand whose dimension is traced out
svd.keep	The optional svd of the kept Kronecker summand
lam.kill	The eigenvalues of the killed Kronecker summand

Details

'KS_dot_vec' computes

matrix((C \oplus A)^i vec(B))

where i is -1 if inv=TRUE and 1 otherwise. The complexity is O((dim(A) + dim(C))^3). If the either svd (of C or A) is provided, the respective cubic term(s) drop out. If, instead, A is diagonal, the complexity simply becomes O(dim(A) dim(C)^2 + dim(C)^3).

'trp_KS' computes

trp((keep \oplus kill)^i)

where i is as above. Here, trp is the partial trace that returns a matrix of the same dimensions as 'keep'. The complexity is O(dim(keep) dim(kill) + dim(keep)^3) if the eigendecomposition of 'keep' is given.

'tr' computes the trace of a matrix.

Value

'KS_dot_vec' returns a matrix of dimension dim(B); 'trp_KS' returns a matrix of dimension dim(keep); 'tr' returns a real number.

Note

There are two partial traces associated with a Kronecker sum–this function computes the partial trace that has the same dimension as the 'keep' matrix.

Author(s)

Andy Dahl

References

Andy Dahl, Victoria Hore, Valentina Iotchkova, Jonathan Marchini (2013). Network inference in matrix-variate Gaussian models with non-independent noise. http://arxiv.org/pdf/1312.1622.pdf

Examples

```
A <- rWishart( 1, 15, diag(7) )[,,1]
B <- matrix( rnorm( 7*11 ), 7, 11 )
C <- rWishart( 1, 15, diag(11) )[,,1]

C_oplus_A <- ( C %x% diag(7) + diag(11) %x% A )

# compare KS_dot_vec to naive implementation
res1  <- KS_dot_vec( A, B, C, inv=TRUE )
res2  <- matrix( solve( C_oplus_A ) %*% c( B ), 7, 11 )
print( all.equal( res1, res2 ) )

# compare trp_KS to naive implementation (on entry [3,4] )
res1  <- trp_KS( C, A )
rows  <- 1:nrow(A) + (3-1)*nrow(A)
cols  <- 1:nrow(A) + (4-1)*nrow(A)
res2  <- tr( solve( C_oplus_A )[ rows, cols ] )
print( all.equal( res1[3,4], res2 ) )
```

LMM *Fit a random effect model*

Description

Fit univarite random effect models and predict missing phenotypes and breeding values.

Usage

```
lmm.impute( Y, K, eig.marg, trace=FALSE )
fit_lmm( y, yprime, K, eig.K, Lam.K, constrained=TRUE )
lmm_obj( delta, yprime, Lam.K )
```

Arguments

Y	Centered and scaled matrix of phenotypes
K	A positive semidefinite kinship matrix describing the genetic covariance between individuals
eig.marg	An optional list containing eigendecompositions of K; entry p eigendecomposes K for the individuals on which phenotype p is observed
trace	Prints algorithm's progress through loop over traits if true
y	Centered and scaled vector of phenotypes
yprime	Kinship-whitened vector of phenotypes
eig.K	The eigendecomposition of K for individuals with observed phenotypes
Lam.K	Eigenvalues of K for individuals with observed phenotypes
delta	Ratio of environmental and genetic variance components
constrained	Whether fitted heritability is constrained to [0,1]

Details

'fit_lmm' fits sig_g and sig_e by maximum likelihood using the model

y ~ N(0, sig_g^2 K) + N(0, sig_e^2 I)

where y is a fully observed vector of phenotypes and K is a known kinship matrix. yprime is a version of y that has been whitened by the eigenvectors of K (i.e. if K = U diag(1/Lam.R) U^T is an eigendecomposition, yprime = U^T y) so that entries of yprime are independent.

'lmm.impute' uses the variance components from univariate lmms (on the observed phenotypes) to predict missing entries of Y and a matrix of breeding values U via BLUPs (best linear unbiased predictions).

Value

'lmm.impute' returns imputed phenotypes, predicted breeding values and variance components for all phenotypes:

`Yhat`	Imputed phenotype matrix
`h2`	Vector of heritibailites for each phenotype
`U`	Predicted matrix of breeding values

'fit_lmm' returns variance components for a single phenotype:

`sig_g`	Genetic variance component
`sig_e`	Environemntal variance component
`h2`	Phenotype heritability

'lmm_obj' returns the log-likelihood.

Author(s)

Andy Dahl

Examples

```
# simulate data
N   <- 500
P   <- 10
K   <- sib_K( N=N )             # indepedendent sets of sibs
Y0  <- rpheno( N=N, P=P, K=K )  # simulate phenotype matrix
Y   <- rmask( Y0, miss=.05 )    # hide phenotypes

# run an lmm on one trait
fit_lmm(y=Y[,1], K=K)

# create+decompose K for observed samples at each phenotype
eig.marg  <- lapply( 1:ncol(Y), function(p){
  obs   <- which( !is.na(Y[,p]) )
  K_loc <- K[ obs, obs ]
  return( eigen( K_loc ) )
})

# fit all lmms and predict phenotypes + breeding values
out   <- lmm.impute( Y, K, eig.marg=eig.marg, trace=TRUE )
```

```
out$h2            # estimated heritabilies
head( out$Yhat )  # imputed phenotypes
```

`misc` *Miscellaneous support functions for phenix*

Description

Miscellaneous support functions for phenix

Usage

```
mat.sqrt( X, inv = FALSE )
schur( X, m )
vec.pinv( x, tol = 1e-08 )
quantnorm( Y )
logdet( X )
```

Arguments

`X`	A symmetric, positive semidefinite matrix
`inv`	Whether to return the inverted square root
`m`	An integer vector of indices defining the Schur complement submatrix
`x`	A numeric vector
`tol`	Value at which entries of x are deemed numerically 0
`Y`	A partially observed matrix

Details

'mat.sqrt' square-roots a covariance matrix by square-rooting (and potentially inverting) its eigenvalues; if any eigenvalues are non-positive (X is not psd), they are set to 1e-8.

'vec.pinv' performs element-wise pseudoinversion; this enables psd matrix pseudoinversion by applying 'vec.pinv' to eigenvalues.

'quantnorm' matches the quantiles of (the observed entries of) columns of Y to standard normal and then scales them to be mean zero and unit variance (which does not automatically hold in the presence of ties).

Value

'mat.sqrt' returns a matrix of dimension dim(X). 'schur' returns the Schur complement of X[m,m] in X, an |m| x |m| matrix. 'vec.pinv' performs element-wise pseudoinversion. 'quantnorm' returns a matrix, of the same dimension and missingness pattern as Y, with quantile-normalized columns. 'logdet' returns a scalar.

Author(s)

Andy Dahl

Examples

```
N     <- 6
P     <- 2
(K    <- sib_K(N,fam_size=3))

mat.sqrt(K)
schur(K, 1:5)
logdet(K)

(Y     <- rpheno(N,P,K))
(Ymiss <- rmask(Y, miss= 0.2))
vec.pinv(Y[,1], tol = 1e-08)
quantnorm(Y)
```

MPMM *Fit a MultiPhenotype Mixed Model*

Description

Fit local maximum likelihood variance components in a multiple phenotype mixed model.

Usage

```
MPMM( Y, K, Y.prime, Lam.R, tol=1e-4, maxit=1e4, warm.updates=TRUE,
       trace=FALSE, print.like=trace,
      lmm_init=TRUE, start.C, start.D, eig.tol=1e-6, standardize=missing(Y.prime) )
```

Arguments

Y	Raw, fully observed phenotype matrix
K	A positive semidefinite kinship matrix describing the genetic covariance between individuals
Y.prime	Kinship pre-whitened phenotype matrix
Lam.R	Eigenvalues of inverse kinship matrix
tol	Convergence criterion
maxit	Maximum number of EM iterations
warm.updates	Initializes M steps with solutions from last iteration if true
trace	Prints output if true
print.like	Includes likelihood evaluations in printout
lmm_init	Initializes C and D with univariate mixed model
start.C, start.D	Optional initalizations
eig.tol	Eigenvalues of K below eig.tol are thresholded to 0
standardize	If TRUE and Y given and unscaled, a warning is produced and Y is scaled internally

Details

This function fits C and D using the model

Y ~ MN(0, K, C^-1) + MN(0, I, D^-1)

where Y is a fully observed matrix of phenotypes, K is a known kinship matrix, and C and D describe the heritable and environmental correlations between phenotypes, respectively. There is also the option to add a variety of penalties to C and D: NA fits the unpenalized model; 'ind' requires the precision matrix to be diagonal; 'iid' requires it to be spherical; 'l1' and 'l2' add lasso ridge penalties, respectively, to all entries of the precision matrix; and 'l1_cor' adds a lasso penalty on the partial correlation matrix.

Y.prime is a version of Y that has been whitened by the eigenvectors of K (i.e. if K = U diag(1/Lam.R) U^T is an eigendecomposition, Y.prime = U^T Y), so that rows of Y.prime are independent.

The algorithm terminates after the relative Frobenius change in C and D is less than reltol or maxit iterations.

Essentially the only effect of the eigenvalue thresholding is to make liklihood computations more stable.

Value

`C`	(Locally) maximum likelihood estimate for genetic precision between phenotypes
`D`	(Locally) maximum likelihood estimate for environmental precision between phenotypes
`h2`	Estimated heritability from C and D
`times`	Cumulative times spent in E-step, M-step for C and M-step for D
`del`	Sequence of relative Frobenius changes in parameters
`niter`	Number of EM iterations
`conv`	Whether EM converged prior to 'maxit' iterations
`C.unscaled,D.unscaled`	C and D on the input Y scale of (if standardize=TRUE and Y is unscaled)
`ll`	Log-likelihood of returned C and D
`ll_path`	Log-likelihood path through EM iterations (if requeested)

Note

The convexity of MPMM is not currently understood, and the output C and D are guaranteed only to be local maximizers of the likelihood.

Author(s)

Andy Dahl

References

Andy Dahl, Victoria Hore, Valentina Iotchkova, Jonathan Marchini (2013). Network inference in matrix-variate Gaussian models with non-independent noise. http://arxiv.org/pdf/1312.1622.pdf

See Also

MPMM_impute, MPMM_helpers

Examples

```
# simulate data
N   <- 1e3
P   <- 7
h2  <- rep( .5, P ) # vector of heritabilities
K   <- sib_K( N=N ) # kinship matrix of indepedendent sets of sibs
Y   <- rpheno( N=N, P=P, K=K, h2=h2 ) # simulate phenotype matrix

# fit an MPMM
out     <- MPMM(scale(Y),K,trace=TRUE,maxit=1e3)
out$h2
```

MPMM_helpers *MPMM helper functions*

Description

The E step, M step and likelihood of the EM algorithm to fit an MPMM and plotting functions

Usage

```
MPMM_Mstep( Omega, pars, warm, init )
MPMM_Estep( C, D, Lam.R, Y.prime )
MPMM_like( Y.prime, Lam.R, C, D )
```

Arguments

Omega	An approximate sample covaiance matrix from E-step
pars	List of penalty parameters; type controls penalty type, rho controls penaltry extent
warm	Uses warm starts if true
init	Warm initialization
C, D	Genetic and environmental precision matrices
Lam.R	Eigenvalues of inverse kinship matrix
Y.prime	Kinship pre-whitened phenotype matrix

Details

'MPMM_Mstep' solves the optimization problem defined in equation (9) of the arxiv paper; this depends on the penalty function used, which is captured by 'pars'. 'MPMM_Estep' evaluates the expected sufficient statistics, given in expressions (7) and (8) in the arxiv paper. 'MPMM_like' evaluates the (unpenalized) likelihood of any 'C' and 'D' pair.

Value

'MPMM_Mstep' returns par, the solution to the penalized likelihood problem defined by the sufficient statistic Omega and input 'pars'. 'MPMM_Estep' returns C and D, estimates of the respective sample covariance, or 'Omega', matrices. 'MPMM_like' returns the likelihood of C and D given data Y.prime and Lam.R.

Author(s)

Andy Dahl

References

Andy Dahl, Victoria Hore, Valentina Iotchkova, Jonathan Marchini (2013). Network inference in matrix-variate Gaussian models with non-independent noise. http://arxiv.org/pdf/1312.1622.pdf

See Also

MPMM, MPMM_impute

`MPMM_impute` *Phenotype imputation with MultiPhenotype Mixed Models*

Description

The method first fits an MPMM on fully-observed samples (rows of Y). Using the resulting variance components, missing entries of Y are imputed to their conditional expectation given the observed entries. If Y is unscaled (has nonzero mean or non-unit variance), it is scaled internally but imputed values are returned on the original scale.

Usage

```
MPMM_impute( Y, K, eig.tol=1e-6, standardize=TRUE, ... )
```

Arguments

`Y`	\# samples x \# phenotypes matrix of partially observed phenotypes
`K`	kinship matrix of relatedness across samples
`eig.tol`	Eigenvalues of K below eig.tol are thresholded to 0
`standardize`	If TRUE and Y is unscaled, a warning is produced and Y is scaled internally
`...`	Arguments passed to MPMM, which is used to fit variance components for BLUPs

Details

Fit an MPMM on fully-observed samples then impute via BLUPs

Value

Y	Imputed phenotype matrix
U	Matrix of estimated breeding values
C	Inverse genetic variance component, estimated by MPMM, used to impute Y
D	Inverse environmental variance component, estimated by MPMM, used to impute Y
VC_out	Output from MPMM run on fully-observed rows of Y

Note

The imputation step is O(N^3 P^3) and, when N is large, can be prohibitively computationally expensive even when fitting the MPMM is cheap.

Author(s)

Andy Dahl

References

Andrew Dahl, Valentina Iotchkova, Amelie Baud, Asa Johansson, Ulf Gyllensten, Nicole Soranzo, Richard Mott, Andreas Kranis \& Jonathan Marchini (2016) A multiple-phenotype imputation method for genetic studies, http://www.nature.com/ng/journal/v48/n4/full/ng.3513.html, Nature Genetics, 48(4), 466-472

See Also

MPMM, MPMM_helpers

Examples

```
# simulation parameters
N   <- 300
P   <- 4
K   <- sib_K( N=N )             # indepedendent sets of sibs
Y0  <- rpheno( N=N, P=P, K=K )  # simulate phenotype matrix
Y   <- rmask( Y0, miss=.05 )    # hide phenotypes

# run MPMM_impute
out <- MPMM_impute(Y,K)
head( out$Y )
```

MVN_impute *Matrix imputation with multivariate normal rows*

Description

An EM algorithm to iteratively impute missing values and estimate the column mean and covariance.

Usage

```
MVN_impute( Y, reltol=1e-4, intercept = TRUE, maxit=1e2, trace=FALSE )
```

Arguments

Y	A partially observed matrix
reltol	The algorithm terminates when the relative mean squared difference between successive imputations is less than reltol
intercept	Whether to fit column means as well as covariances
maxit	Maximum number of EM iterations
trace	Whether to print updates after each EM iteration

Details

This models rows of Y as i.i.d. N(mu, Sigma). The EM algorithm iteratively predicts the sufficient statistics and maximizes the resulting likelihood, ultimately returning imputed values for missing entries in Y and MLEs for mu and Sigma. If 'intercept'=FALSE, mu is constrained to 0.

Value

Y	The imputed matrix
Sigma	The estimated covariance across columns of Y

Author(s)

Andy Dahl

References

Andrew Dahl, Valentina Iotchkova, Amelie Baud, Asa Johansson, Ulf Gyllensten, Nicole Soranzo, Richard Mott, Andreas Kranis \& Jonathan Marchini (2016) A multiple-phenotype imputation method for genetic studies, http://www.nature.com/ng/journal/v48/n4/full/ng.3513.html, Nature Genetics, 48(4), 466-472

Examples

```
# simulate data
N   <- 1e3
P   <- 10
Y0  <- matrix( rnorm( N*P ), N, P )
Y   <- rmask( Y0, miss=.05 )

# impute
out <- MVN_impute( Y, trace=TRUE )
head( out$Y )
```

mv_gwas *Perform multiphenotype GWAS via approximate LRT*

Description

Approximate multivariate GWAS method to test whether each of L SNPs is associated with (at least one of) the P traits.

Usage

```
mv_gwas( Y, G, K, Y.prime, G.prime, lam_K, trace=FALSE, eig.tol=1e-6, ... )
alt_ll_fxn( Y.prime, G.prime, C, D, lam_K, trace=FALSE )
```

Arguments

Y	Fully observed phenotype matrix
G	Genotype matrix with SNPs as columns
K	A positive semidefinite kinship matrix describing the genetic covariance between individuals
Y.prime	Kinship pre-whitened phenotype matrix
G.prime	Scaled, then kinship pre-whitened genotype matrix
lam_K	Eigenvalues of the kinship matrix K
trace	Prints progress through the algorithm if true
eig.tol	Eigenvalues of K below eig.tol are threshholded to 0
...	Arguments passed to MPMM, used to fit VCs under the null
C,D	Inverse variance components from fitting MPMM under the null

Details

For each vector of SNPs g, or column of G, these functions fit the model

Y ~ g beta + MN(0, K, C^-1) + MN(0, I, D^-1)

where Y is a fully observed matrix of phenotypes, K is a known kinship matrix, and C and D are precisions describing the heritable and environmental correlations between phenotypes, respectively. The LRTs test the hypothesis that beta != 0; the approximation is that C and D are fit only once, under the null that beta=0, rather than fit jointly with beta at each SNP.

The algorithm requires only Y.prime, G.prime and lam_K; if any of these are missing, however, then Y, G and K must all be provided so the former can be computed.

Value

'alt_ll_fxn' returns the (approximate) maximum likelihood under the alternatives.

'mv_gwas' returns a list:

lrt	Vector of LRT statistics for each SNP in G
p	Vector of LRT p-values for each SNP in G
null	Result of running 'MPMM' under the null

Author(s)

Andy Dahl

References

Andrew Dahl, Valentina Iotchkova, Amelie Baud, Asa Johansson, Ulf Gyllensten, Nicole Soranzo, Richard Mott, Andreas Kranis \& Jonathan Marchini (2016) A multiple-phenotype imputation method for genetic studies, http://www.nature.com/ng/journal/v48/n4/full/ng.3513.html, Nature Genetics, 48(4), 466-472

See Also

MPMM

Examples

```
# dummy data
N   <- 200
P   <- 3
Y   <- matrix( rnorm(N*P), N, P )
y   <- c(Y)
g   <- as.numeric(scale(rbinom( N, 2, .5 )))
K   <- rWishart( 1, N, diag(N) )[,,1]
C   <- rWishart( 1, P, diag(P) )[,,1]
D   <- rWishart( 1, P, diag(P) )[,,1]
eig.K <- eigen(K)
Q     <- eig.K$vec
lam_K <- eig.K$val

# fast log-like
ll_fast <- alt_ll_fxn( Y.prime=t(Q)%*%Y, G.prime=t(Q)%*%g, C, D, lam_K=lam_K )

# check against naive computation:
# first compute betahat then evaluate ll

# two slow ways to compute beta:
# first:
Sigma <- solve(C) %x% K + solve(D) %x% diag(N)
trans <- mat.sqrt( Sigma, inv=TRUE )
X     <- diag(P) %x% matrix(g,N,1)
yt    <- trans %*% y
Xt    <- trans %*% X
betahat1 <- as.numeric( coef( lm( yt ~ Xt-1 ) ) )

# second:
Omega <- solve(Sigma)
betahat  <- as.numeric( solve( t(X) %*% Omega %*% X ) %*% t(X) %*% Omega %*% y )
all.equal( betahat, betahat1 )

# ll evaluation using betahat
ldet  <- -logdet( Sigma )
resid <- c( Y - g %o% betahat )
quad  <- -as.numeric( t( resid ) %*% solve( Sigma ) %*% resid )
all.equal( ll_fast, ll_slow  <- 1/2*( ldet + quad ) )
```

```
# run multiphenotype 'GWAS' on 2 simulated SNPs
G   <- cbind( g, scale(rbinom( N, 2, .5 )) )  # 'genome-wide'
out <- mv_gwas( Y=Y, G=G, K=K, maxit=1e3 )    # 'GWAS'
out$lrt
out$null$h2
```

---

phenix *PHENotype Imputation eXpedited*

Description

Imputes missing entries of a phenotype matrix and predicts breeding values

Usage

```
phenix( Y, K, Q, lam_K,
    test=FALSE, test.frac=.05, test.cols=1:ncol(Y),
    seed=8473, quantnorm=FALSE, scale=TRUE,
    trim=FALSE, trim.sds=4, ... )
```

Arguments

Y	A partially observed matrix of phenotypes
K	A positive semidefinite kinship matrix describing the genetic covariance between individuals
Q	Optional eigenvectors of the kinship matrix
lam_K	Optional eigenvalues of the kinship matrix
test	Option to test phenix's performance on a specific dataset by hiding some entries of Y and assessing recovery accuracy
test.frac	The fraction of Y's observed entries hidden in the testing procedure
test.cols	Which columns of Y can be masked in the testing procedure
seed	Enables reproducibility across test runs
quantnorm	Quantile normalizes phenotypes prior to imputation
scale	Whether to center and standardize columns of Y; this process is inverted after imputation so imputed phenotypes are on the input scale
trim	Whether to remove outliers, defined column-wise as entries more than 'trim.sds' standard deviations from their means
trim.sds	Number of standard deviations used to define outliers
...	Arguments passed to phenix_vb, mainly 'tau', which allows additional shrinkage of U, and 'maxit' and 'reltol', which define convergence

Details

phenix incorporates genetic relatedness into a Bayesian matrix factorization model to impute missing phenotypes in Y. It computes a low-rank reconstruction of U, the genetic contribution to phenotypic variance. It also estimates B and E, genetic and environmental phenotypic correlation matrices, from U and the environmental residual Y-U, respectively.

Value

imp	Imputed phenotype matrix
U	Matrix of estimated breeding values
S	Matrix of latent components
h2	Vector of estimated phenotype heritabilities
B	Estimated genetic covariance matrix between phenotypes
E	Estimated environmental covariance matrix between phenotypes
Q	Eigenvectors of K
lam_K	Eigenvalues of K
vb_out	Output from phenix_vb, the workhorse function
outliers	The entries of 'c(Y)' set to NA prior to imputation
test	If test=TRUE, a list containing: 'time', the runtime; 'cors' and 'cor_glob', column-wise and overall imputation correlations; and 'mses' and 'mse_glob', column-wise and overall imputation mean-squared errors

Author(s)

Andy Dahl

References

Andrew Dahl, Valentina Iotchkova, Amelie Baud, Asa Johansson, Ulf Gyllensten, Nicole Soranzo, Richard Mott, Andreas Kranis \& Jonathan Marchini (2016) A multiple-phenotype imputation method for genetic studies, http://www.nature.com/ng/journal/v48/n4/full/ng.3513.html, Nature Genetics, 48(4), 466-472

See Also

sim_data,phenix_vb

Examples

```
# simulate data
N   <- 500
P   <- 20
K   <- sib_K( N=N )               # indepedendent sets of sibs
Y0  <- rpheno( N=N, P=P, K=K )    # simulate phenotype matrix
Y   <- rmask( Y0, miss=.05 )      # hide phenotypes

out     <- phenix(Y,K,test=TRUE)  # impute missing data
sub   <- which( is.na(Y) )
cor( out$imp[ sub ], Y0[ sub ] )  #true overall accuracy
out$test$cor_glob                 #estimated accuracies
```

phenix_vb *Variational Bayes algorithm to fit phenix*

Description

Workhorse functions for phenix

Usage

```
phenix_vb( Y, Q, lam_K, N=nrow(Y), P=ncol(Y), M=min(c(N,P)),
    tau=0, e=P+5, E.inv=solve( diag(P)/(e-P-1) ),
    trace=0, reltol=1e-8, maxit=1e3, cutoff=1e-3, ... )
phenix_likelihood( Y, Q, lam_K, N, P, M, tau, E.inv,
    mu_Y, Sigma_Y, Omega, mu_beta, mu_S, eprime,
    svd.Omega, svd.V_S, svd.V_beta, svd.Omega_beta,
    j2miss, j2len, n.miss.types )
```

Arguments

Y	A matrix of phenotypes
Q	Eigenvectors of the kinship matrix
lam_K	Eigenvalues of the kinship matrix
N	Number of samples, or rows of Y
P	Number of phenotypes, or columns of Y
M	Maximum allowed rank of the low-rank, genetic term
tau	The prior precision on beta. Increasing tau increases shrinkage of the low-rank, genetic term
e	Weight of Wishart prior on environmental covariance; the prior is scaled to always have expectation diag('P')
E.inv	Inverse shape parameter of Wishart prior on environmental covariance
trace	Whether to print output at each VB iteration
reltol	Convergence criterion for relative change in the approximate log-likelihood
maxit	Maximum number of VB iterations
cutoff	Cutoff for component to be deemed zero
mu_Y, Sigma_Y, Omega, mu_beta, mu_S, eprime	Variational parameters
svd.Omega, svd.V_S, svd.V_beta, svd.Omega_beta	SVD's of variational parameters
j2miss, j2len, n.miss.types	Compressed representation of missingness pattern in Y
...	Arguments passed to 'MVN_impute', used for initialization

Details

'phenix_vb' is a deterministic algorithm that iteratively updates blocks of the variational parameters for the approximate posterior. The chosen variational approximation admits analytic solutions for each of these block updates; the algorithm iterates these updates (a.k.a. iterated conditional modes) until the relative change in the marginal likelihood lower bound (a.k.a. 'phenix_likelihood') is below 'reltol'.

'phenix_likelihood' is the objective function for 'phenix_vb'; maximizing this likelihood lower bound is equivalent to minimizing the KL-divergence between the exact posterior and its variational approximation. The VB algorithm is guaranteed to increase this likelihood at each iteration, and thus returns a local maximum of 'phenix_likelihood'.

Value

'phenix_vb' returns a locally optimal set of variational parameters for the approximation to the phenix posterior. 'phenix_likelihood' evaluates the marginal likelihood lower bound given by a set of variational parameters.

Note

These functions should rarely be called directly; the function 'phenix' should be used instead.

Author(s)

Andy Dahl

References

Andrew Dahl, Valentina Iotchkova, Amelie Baud, Asa Johansson, Ulf Gyllensten, Nicole Soranzo, Richard Mott, Andreas Kranis \& Jonathan Marchini (2016) A multiple-phenotype imputation method for genetic studies, http://www.nature.com/ng/journal/v48/n4/full/ng.3513.html, Nature Genetics, 48(4), 466-472

See Also

phenix

`sim_data` *Simulate phenix data*

Description

Simulates kinship matrix and partially observed phenotype matrix

Usage

```
sib_K( N, fam_size=5 )
rpheno( N, P, K, h2=rep( .5, P ), B, E )
rmask( Y0, miss=.05 )
```

Arguments

N	Number of desired samples
fam_size	Size of each family in kinship matrix
P	Number of desired traits
K	A positive semidefinite kinship matrix describing the relatedness between samples
h2	A length 'P' vector of phenotype heritabilities
B	An optional positive semidefinite matrix of genetic correlations; defaults to Wishart(P, 1/P*diag(P))
E	An optional positive semidefinite matrix of environmental correlations; defaults to Wishart(P, 1/P*diag(P))
Y0	A fully-observed matrix of phenotypes
miss	The fraction of entries in Y0 to be set to missing

Details

These functions simulate the necessary input data for phenix from the MPMM model; see the MPMM manual page for a definition of this model and the phenix manual page for examples. sib_K generates the matrix of relatedness between samples; rpheno generates heritable traits based on K and the between-trait covariances B and E; rmask then sets some of these entries to missing completely-at-random.

Value

All functions simulate input data for phenix. 'sib_K' returns a block diagonal NxN kinship matrix with blocks corresponding to independent sets of siblings; 'rpheno' returns an NxP matrix of phenotypes; and 'rmask' hides entries in a matrix completely-at-random.

Author(s)

Andy Dahl

See Also

MPMM,phenix

Examples

```
N       <- 6
P       <- 2
(K      <- sib_K(N,fam_size=3))
(Y      <- rpheno(N,P,K))
(Ymiss  <- rmask(Y, miss= 0.2))
```

Index

∗Topic **algebra**
kronsum, 1
LMM, 3
MPMM, 6
MPMM_helpers, 8

alt_ll_fxn *(*mv_gwas*)*, 12

fit_lmm *(*LMM*)*, 3

kronsum, 1
KS_dot_vec *(*kronsum*)*, 1

lik.fxn *(*MPMM*)*, 6
LMM, 3
lmm.impute *(*LMM*)*, 3
lmm_obj *(*LMM*)*, 3
logdet *(*misc*)*, 5

mat.sqrt *(*misc*)*, 5
misc, 5
MPMM, 6, *9, 10*, *13*, *18*
MPMM_Estep *(*MPMM_helpers*)*, 8
MPMM_helpers, *8*, 8, *10*
MPMM_impute, *8, 9*, 9
MPMM_like *(*MPMM_helpers*)*, 8
MPMM_Mstep *(*MPMM_helpers*)*, 8
mv_gwas, 12
MVN_impute, 10

phenix, 14, *17, 18*
phenix_likelihood *(*phenix_vb*)*, 16
phenix_vb, *15*, 16

quantnorm *(*misc*)*, 5

rmask *(*sim_data*)*, 17
rpheno *(*sim_data*)*, 17

schur *(*misc*)*, 5
sib_K *(*sim_data*)*, 17
sim_data, *15*, 17

tr *(*kronsum*)*, 1
trp_KS *(*kronsum*)*, 1

vec.pinv *(*misc*)*, 5