Package 'SAMM'

October 12, 2022

Type Package

Title Some Algorithms for Mixed Models

Version 1.1.1

Date 2018-11-22

Author Deniz Akdemir

Maintainer Deniz Akdemir <deniz.akdemir.work@gmail.com>

Description This program can be used to fit Gaussian linear mixed models (LMM). Univariate and multivariate response models, multiple variance components, as well as, certain correlation and covariance structures are supported. In many occasions, the user can pick one of the several mixed model fitting algorithms, which are explained further in the details section. Some algorithms are specific to certain types of models (univariate or multivariate, diagonal or non-diagonal residual, one or multiple variance components, etc,...).

License GPL-3

Imports Rcpp (>= 0.12.4)

LinkingTo Rcpp, RcppArmadillo

NeedsCompilation yes

Repository CRAN

Date/Publication 2018-12-06 16:30:02 UTC

R topics documented:

SAMM-package *Some Algorithms for Mixed Models*

Description

This program can be used to fit Gaussian linear mixed models (LMM). Univariate and multivariate response models, multiple variance components, as well as, certain correlation and covariance structures are supported. In many occasions, the user can pick one of the several mixed model fitting algorithms, which are explained further in the details section. Some algorithms are specific to certain types of models (univariate or multivariate, diagonal or non-diagonal residual, one or multiple variance components, etc,...).

Details

The DESCRIPTION file:

Package:	SAMM
Type:	Package
Title:	Some Algorithms for Mixed Models
Version:	1.1.1
Date:	2018-11-22
Author:	Deniz Akdemir
Maintainer:	Deniz Akdemir <deniz.akdemir.work@gmail.com>
Description:	This program can be used to fit Gaussian linear mixed models (LMM). Univariate and multivariate response mo
License:	GPL-3
Imports:	Rcpp (>= 0.12.4)
LinkingTo:	Rcpp, RcppArmadillo

Index of help topics:

This program can be used to fit Gaussian linear mixed models (LMM). Univariate and multivariate response models, multiple variance components, as well as, certain correlation and covariance structures are supported. In many occations, the user can pick one of the several mixed model fitting algorithms, which are explained further in the details section. Some algorithms are specific to certain types of models (univariate or multivariate, diagonal or nondiagonal residual, one or multiple variance components, etc,...).

Author(s)

Deniz Akdemir

Maintainer: Deniz Akdemir <deniz.akdemir.work@gmail.com>

References

Bates, Douglas M. "lme4: Mixed-effects modeling with R." URL http://lme4. r-forge. r-project. org/book (2010).

Zhou, Hua, et al. "MM Algorithms for Variance Components Models." arXiv preprint arXiv:1509.07426 (2015).

Zhou, Xiang, and Matthew Stephens. "Efficient algorithms for multivariate linear mixed models in genome-wide association studies." Nature methods 11.4 (2014): 407.

Kang, Hyun Min, et al. "Efficient control of population structure in model organism association mapping." Genetics 178.3 (2008): 1709-1723.

Gilmour, Arthur R., Robin Thompson, and Brian R. Cullis. "Average information REML: an efficient algorithm for variance parameter estimation in linear mixed models." Biometrics (1995): 1440-1450.

See Also

~~ Optional links to other man pages, e.g. ~~

Examples

```
x=2
```

`loglikfuncmmmkmv` *Calculate the loglikeligood for a general mixed model*

Description

Calculate the loglikeligood for the LMM model expressed as

$$Y = XB + \sum_{j=1}^{k} Z_j G_j + WE,$$

where Y is the $n \times d$ response variable, X is the $n \times q$ design matrix of $q \times d$ the fixed effects B, Z_j for $j = 1, 2, \ldots, k$ ($k \geq 1$) are the $n \times q_j$ design matrices of the $q_j \times d$ random effects G_j, and W is the $n \times t$ design matrix of $t \times d$ residual effecs E. The random effects and the residual are independently distributed, and have matrix variate distributions ($G_j \sim N_{q_j \times d}(0_{q_j \times d}, K_j, \Sigma_j)$ for $j = 1, 2, \ldots, k$ and $E \sim N_{t \times d}(0_{t \times d}, R, \Sigma_E)$).

Usage

```
loglikfuncmmmkmv(Y,X,Zlist, Klist, sigmahatlist, B,W,R )
```

Arguments

Y	a numeric vector for the parameters (a mapping of the original parameters)
X	a numeric matrix (see examples and details)
Zlist	a numeric matrix (see examples and details)
Klist	a numeric matrix (see examples and details)
sigmahatlist	a numeric matrix (see examples and details)
B	bla bla
W	bla bal
R	bla bla

Details

bla bla

Value

bla bla

Author(s)

Deniz Akdemir

Examples

```
## Not run:
library(SAMM)
n=100
nsample=80
rhotrans=5
ar1cov_cppforR(c(rhotrans),matrix(5))
rho=(2/pi)*atan(rhotrans)
rho
tan((pi/2)*(rho))

M1<-matrix(rbinom(n*300, 2, .2)-1, nrow=n)
K1<-relmatcov_cppforR(c(.01), M1)

M2<-matrix(rbinom(n*300, 2, .2)-1, nrow=n)
K2<-relmatcov_cppforR(c(0.03), M2)
W=(diag(5)[sample(1:5,n, replace=TRUE),])
covY<-3*K1+5*K2+10*(W%*%ar1cov_cppforR(c(rhotrans),matrix(5))%*%t(W))
K1[1:5,1:5]
dim(W)
dim(ar1cov_cppforR(c(6),matrix(5)))
Y<-10+crossprod(chol(covY),rnorm(n))


#training set
Trainset<-sample(1:n,nsample)
```

```
ytrain=Y[Trainset]
Xtrain=matrix(rep(1, n)[Trainset], ncol=1)
Ztrain=diag(n)[Trainset,]
Wtrain=W[Trainset,]

samout<-SAMM(Y=matrix(ytrain,ncol=1),X=Xtrain,
Zlist=list(Ztrain, Ztrain), Klist=list(K1,K2),
lambda=0, W=Wtrain,R=list("ar1",c(0),matrix(5,1,1)),
Siglist=list("","",""), corfunc=c(F,F,T), corfuncfixed=c(F,F,F),
sigfunc=c(F,F,F),mmalg="dermm_reml2", tolparconv=1e-10,
tolparinv=1e-10,maxiter=1000,geterrors=F)
samout$corfuncparamslist[[3]]
rhohat=(2/pi)*atan(samout$corfuncparamslist[[3]])
rhohat
ar1cov_cppforR(c(samout$corfuncparamslist[[3]]),matrix(5,1,1))

## End(Not run)
```

SAMM *Some Algoritms for Mixed Models*

Description

The LMM model of focus can be expressed as

$$Y = XB + \sum_{j=1}^{k} Z_j G_j + WE,$$

where Y is the $n \times d$ response variable, X is the $n \times q$ design matrix of $q \times d$ the fixed effects B, Z_j for $j = 1, 2, \ldots, k$ ($k \geq 1$) are the $n \times q_j$ design matrices of the $q_j \times d$ random effects G_j, and W is the $n \times t$ design matrix of $t \times d$ residual effecs E. The random effects and the residual are independently distributed, and have matrix variate distributions ($G_j \sim N_{q_j \times d}(0_{q_j \times d}, K_j, \Sigma_j)$ for $j = 1, 2, \ldots, k$ and $E \sim N_{t \times d}(0_{t \times d}, R, \Sigma_E)$). The matrices $K_j, R, \Sigma_j, \Sigma_E$ might be further parametrized. When the response is univariate $(d = 1)$ there is no need to specify a structure for Σ_j's andΣ_E. On the other hand, the K_j and R are kernel matrices (covariance matrices with some standardization over the diagonals) and they need to be provided by the user.

Please refer to the examples below and the other help files for more details about the kernel and covariance structures.

Usage

```
SAMM(Y, X, Zlist, Klist, lambda, W, R, Siglist, corfunc,
                 corfuncfixed, sigfunc, mmalg, tolparconv = 1e-10,
                 tolparinv = 1e-10, maxiter = 1000L, geterrors = FALSE,
                 Hinv = FALSE)
```

Arguments

`Y`	Y is the $n \times d$ response variable
`X`	$n \times q$ design matrix of $q \times d$ the fixed effects B
`Zlist`	a list object containing Z_j for $j = 1, 2, \ldots, k$ ($k \geq 1$), the $n \times q_j$ design matrices of the $q_j \times d$ random effects G_j,
`Klist`	a list to specify the kernel matrices K_j for $j = 1, 2, \ldots, k$. For each j, the user needs to provide a constant matrix, or a list specifying the kernel structure
`lambda`	a scalar shrinkage parameter for shrinkage of variance components (only works with the choice mmalg="mmmk_ml").
`W`	W is the $n \times t$ design matrix of $t \times d$ residual effecs E
`R`	a list to specify the kernel matrix R,the user needs to provide a constant matrix, or a list specifying the kernel structure
`Siglist`	a list to specify the covariance structures Σ_j's andΣ_E
`corfunc`	a boolian vector specifying whether K_j for $j = 1, 2, \ldots, k$ and R are functions or given matrices (TRUE for functions)
`corfuncfixed`	a boolian vector specifying whether K_j for $j = 1, 2, \ldots, k$ and R are fixed at the initial parameter values specified
`sigfunc`	a boolian vector specifying whether Σ_j for $j = 1, 2, \ldots, k, E$ are functions or unstructured (TRUE for functions)
`mmalg`	The mixed model solving algorithm
`tolparconv`	convergence criteria
`tolparinv`	a small scalar to add to the diagonals of positive semidefinite matrices for inversion or for calculating the Cholesky decompositions
`maxiter`	Maximum number of iterations
`geterrors`	TRUE or FALSE, if true prediction error variances for the random effects are supplied in the output
`Hinv`	TRUE or FALSE, if TRUE the inverse of the

Details

This might change with respect to the algorithm or analysis.

Value

This might change with respect to the algorithm or analysis.

Author(s)

Deniz Akdemir

Examples

```
## Not run:
library(SAMM)
#set.seed(12345)

n=120
ntrain=100
M1<-matrix(rbinom(n*180,2,.3)-1, nrow=n)
K<-relmatcov_cppforR(c(0), M1)
K[1:5,1:5]
det(K)
K=K+1e-3*diag(n)
mean(diag(K))

covY<-2*K+1*diag(n)

Y<-10+crossprod(chol(covY),rnorm(n))


#training set
Trainset<-sample(1:n,ntrain,replace=(ntrain>n))
y=Y[Trainset]
X=matrix(rep(1, n)[Trainset], ncol=1)
Z=diag(n)[Trainset,]

X=X
y=y



########1
samout1<-SAMM(Y=matrix(y,ncol=1),X=X, Zlist=list(Z), Klist=list(K),
lambda=0, W=diag(ntrain),R=list(diag(ntrain)),Siglist=list(),
corfunc=c(F,F), corfuncfixed=c(F,F), sigfunc=c(F),mmalg="emm_ml",
tolparconv=1e-10, tolparinv=1e-10,maxiter=1000,geterrors=F)

samout2<-SAMM(Y=matrix(y,ncol=1),X=X, Zlist=list(Z), Klist=list(K),
lambda=0, W=diag(ntrain),R=list(diag(ntrain)),Siglist=list(),
corfunc=c(F,F), corfuncfixed=c(F,F), sigfunc=c(F),mmalg="emm_reml",
tolparconv=1e-10, tolparinv=1e-10,maxiter=1000,geterrors=F)

samout3<-SAMM(Y=matrix(y,ncol=1),X=X, Zlist=list(Z),
Klist=list(list("Const",c(0),K)), lambda=0,
W=diag(ntrain),R=list(diag(ntrain)),Siglist=list(),
corfunc=c(F,F), corfuncfixed=c(F,F), sigfunc=c(F),
mmalg="dmm_ml", tolparconv=1e-10, tolparinv=1e-10,
maxiter=1000,geterrors=F)

samout4<-SAMM(Y=matrix(y,ncol=1),X=X, Zlist=list(Z),
Klist=list(list("Const",c(0),K)), lambda=0,
W=diag(ntrain),R=list(diag(ntrain)),Siglist=list(),
corfunc=c(F,F), corfuncfixed=c(F,F), sigfunc=c(F),
```

```
mmalg="dmm_reml", tolparconv=1e-10, tolparinv=1e-10,
maxiter=1000,geterrors=F)

samout5<-SAMM(Y=matrix(y,ncol=1),X=X, Zlist=list(Z),
Klist=list(K), lambda=0, W=diag(ntrain),
R=list(diag(ntrain)),Siglist=list(), corfunc=c(F,F),
corfuncfixed=c(F,F), sigfunc=c(F),mmalg="mm_ml",
tolparconv=1e-10, tolparinv=1e-10,maxiter=1000,geterrors=F)

samout6<-SAMM(Y=matrix(y,ncol=1),X=X, Zlist=list(Z),
Klist=list(K), lambda=0, W=diag(ntrain),R=list(diag(ntrain)),
Siglist=list(), corfunc=c(F,F), corfuncfixed=c(F,F),
sigfunc=c(F),mmalg="dermm_reml1", tolparconv=1e-10,
tolparinv=1e-10,maxiter=1000,geterrors=F)

samout7<-SAMM(Y=matrix(y,ncol=1),X=X, Zlist=list(Z),
Klist=list(K), lambda=0, W=diag(ntrain),R=list(diag(ntrain)),
Siglist=list(), corfunc=c(F,F), corfuncfixed=c(F,F),
sigfunc=c(F),mmalg="dermm_reml2", tolparconv=1e-10,
tolparinv=1e-10,maxiter=1000,geterrors=F)

samout8<-SAMM(Y=matrix(y,ncol=1),X=X, Zlist=list(Z),
Klist=list(K), lambda=0, W=diag(ntrain),R=list(diag(ntrain)),
Siglist=list(), corfunc=c(F,F), corfuncfixed=c(F,F),
sigfunc=c(F),mmalg="mmmk_ml", tolparconv=1e-10,
tolparinv=1e-10,maxiter=1000,geterrors=F)

samout9<-SAMM(Y=matrix(y,ncol=1),X=X, Zlist=list(Z),
Klist=list(K), lambda=0, W=diag(ntrain),R=list(diag(ntrain)),
Siglist=list(), corfunc=c(F,F), corfuncfixed=c(F,F),
sigfunc=c(F),mmalg="emmmk_reml", tolparconv=1e-10,
tolparinv=1e-10,maxiter=1000,geterrors=F)

samout10<-SAMM(Y=matrix(y,ncol=1),X=X, Zlist=list(Z),
Klist=list(K), lambda=0, W=diag(ntrain),R=list(diag(ntrain)),
Siglist=list(), corfunc=c(F,F), corfuncfixed=c(F,F),
sigfunc=c(F),mmalg="emmmk_ml", tolparconv=1e-10,
tolparinv=1e-10,maxiter=1000,geterrors=F)

samout11<-SAMM(Y=matrix(y,ncol=1),X=X, Zlist=list(Z),
Klist=list(K), lambda=0, W=diag(ntrain),R=list(diag(ntrain)),
Siglist=list(), corfunc=c(F,F), corfuncfixed=c(F,F), sigfunc=c(F),
mmalg="emmmv_ml", tolparconv=1e-10, tolparinv=1e-10,maxiter=1000,geterrors=F)

###########2
R<-diag(ntrain)
diag(R)<-diag(R)+.01*rnorm(nrow(R))

samout12<-SAMM(Y=matrix(y,ncol=1),X=X, Zlist=list(Z),
Klist=list(K), lambda=0, W=diag(ntrain),R=list(R),
Siglist=list(), corfunc=c(F,F), corfuncfixed=c(F,F),
sigfunc=c(F),mmalg="mmmk_ml", tolparconv=1e-10,
tolparinv=1e-10,maxiter=1000,geterrors=F)
```

```
samout13<-SAMM(Y=matrix(y,ncol=1),X=X, Zlist=list(Z),
Klist=list(K), lambda=0, W=diag(ntrain),R=list(R),
Siglist=list(), corfunc=c(F,F), corfuncfixed=c(F,F),
sigfunc=c(F),mmalg="mm_ml", tolparconv=1e-10,
tolparinv=1e-10,maxiter=1000,geterrors=F)

samout14<-SAMM(Y=matrix(y,ncol=1),X=X, Zlist=list(Z),
Klist=list(K), lambda=0, W=diag(ntrain),R=list(R),
Siglist=list(), corfunc=c(F,F), corfuncfixed=c(F,F),
sigfunc=c(F),mmalg="mmmv_ml", tolparconv=1e-10,
tolparinv=1e-10,maxiter=1000,geterrors=F)

samout15<-SAMM(Y=matrix(y,ncol=1),X=X, Zlist=list(Z),
Klist=list(K), lambda=0, W=diag(ntrain),R=list(R),
Siglist=list(), corfunc=c(F,F), corfuncfixed=c(F,F),
sigfunc=c(F),mmalg="mmmkmv_ml", tolparconv=1e-10,
tolparinv=1e-10,maxiter=1000,geterrors=F)

samout16<-SAMM(Y=matrix(y,ncol=1),X=X, Zlist=list(Z),
Klist=list(K), lambda=0, W=diag(ntrain),R=list(R),Siglist=list(),
corfunc=c(F,F), corfuncfixed=c(F,F), sigfunc=c(F),
mmalg="dermm_reml1", tolparconv=1e-10,
tolparinv=1e-10,maxiter=1000,geterrors=F)

samout17<-SAMM(Y=matrix(y,ncol=1),X=X, Zlist=list(Z),
Klist=list(K), lambda=0, W=diag(ntrain),R=list(R),
Siglist=list(), corfunc=c(F,F), corfuncfixed=c(F,F),
sigfunc=c(F),mmalg="dermm_reml2", tolparconv=1e-10,
tolparinv=1e-10,maxiter=1000,geterrors=F)

samout18<-SAMM(Y=matrix(y,ncol=1),X=X, Zlist=list(Z),
Klist=list(list("Const",c(0),K)), lambda=0, W=diag(ntrain),
R=list(R),Siglist=list(), corfunc=c(F,F),
corfuncfixed=c(F,F), sigfunc=c(F),mmalg="dmm_ml",
tolparconv=1e-10, tolparinv=1e-10,maxiter=1000,geterrors=F)

samout19<-SAMM(Y=matrix(y,ncol=1),X=X, Zlist=list(Z),
Klist=list(list("Const",c(0),K)), lambda=0, W=diag(ntrain),
R=list(R),Siglist=list(), corfunc=c(F,F), corfuncfixed=c(F,F),
sigfunc=c(F),mmalg="dmm_reml", tolparconv=1e-10,
tolparinv=1e-10,maxiter=1000,geterrors=F)



##3
samout20<-SAMM(Y=matrix(y,ncol=1),X=X, Zlist=list(Z),
Klist=list(list("Const",c(0),K)), lambda=0, W=diag(ntrain),
R=list(R),Siglist=list(), corfunc=c(T,F), corfuncfixed=c(F,F),
sigfunc=c(F),mmalg="dmm_reml", tolparconv=1e-10,
tolparinv=1e-10,maxiter=1000,geterrors=F)
```

```
samout21<-SAMM(Y=matrix(y,ncol=1),X=X, Zlist=list(Z),
Klist=list(list("ar1",c(0),matrix(n,1,1))), lambda=0,
W=diag(ntrain),R=list(R),Siglist=list(), corfunc=c(T,F),
corfuncfixed=c(F,F), sigfunc=c(F),mmalg="dmm_reml",
tolparconv=1e-10, tolparinv=1e-10,maxiter=1000,geterrors=F)


samout22<-SAMM(Y=matrix(y,ncol=1),X=X, Zlist=list(Z),
Klist=list(list("ar1het",rep(0,n),matrix(n,1,1))), lambda=0,
W=diag(ntrain),R=list(R),Siglist=list(), corfunc=c(T,F),
corfuncfixed=c(F,F), sigfunc=c(F),mmalg="dmm_reml",
tolparconv=1e-10, tolparinv=1e-10,maxiter=1000,geterrors=F)

samout1$Vu
samout2$Vu
samout3$Vu
samout4$Vu
samout5$Vu
samout6$Vu
samout7$Vu
samout8$Vu
samout9$Vu
samout10$Vu
samout11$Vu

samout12$Vu
samout13$Vu
samout14$sigmahatlist[[1]]
samout15$sigmahatlist[[1]]
samout16$Vu
samout17$Vu
samout18$Vu
samout19$Vu

samout20$Ve
str(samout19)


str(samout20)
str(samout21)
str(samout22)




###

samout1$Vu
samout2$Vu
samout3$Vu
samout4$Vu
samout5$Vu
samout6$Vu
```

```
samout7$Vu
samout8$Vu
samout9$Vu
samout10$Vu
samout11$Vgt

samout12$Vu
samout13$Vu
samout14$sigmahatlist[[1]]
samout15$sigmahatlist[[1]]
samout16$Vu
samout17$Vu
samout18$Vu
samout19$Vu

samout20$Ve
str(samout19)


str(samout20)
str(samout21)
str(samout22)



##############################


n=100
nsample=80
rhotrans=5
rho=tan((pi/2)*print((2/pi)*atan(rhotrans)))
rho
M1<-matrix(rbinom(n*300, 2, .2)-1, nrow=n)
K1<-relmatcov_cppforR(c(.01), M1)

M2<-matrix(rbinom(n*300, 2, .2)-1, nrow=n)
K2<-relmatcov_cppforR(c(-0.3), M2)
##K2<-ar1_cppforR(c(20), matrix(n))
W=(diag(5)[sample(1:5,n, replace=T),])
covY<-3*K1+5*K2+10*(W%*%ar1cov_cppforR(c(rhotrans),matrix(5))%*%t(W))
K1[1:5,1:5]
dim(W)
dim(ar1cov_cppforR(c(6),matrix(5)))
Y<-10+crossprod(chol(covY),rnorm(n))


#training set
Trainset<-sample(1:n,nsample)
ytrain=Y[Trainset]
Xtrain=matrix(rep(1, n)[Trainset], ncol=1)
Ztrain=diag(n)[Trainset,]
Wtrain=W[Trainset,]
```

```
samout27<-SAMM(Y=matrix(ytrain,ncol=1),X=Xtrain,
Zlist=list(Ztrain, Ztrain), Klist=list(K1,K2),
lambda=0, W=Wtrain,R=list(list("ar1",c(0),matrix(5,1,1))),
Siglist=list("","",""), corfunc=c(F,F,T),
corfuncfixed=c(F,F,F), sigfunc=c(F,F,F),mmalg="mmmk_ml",
tolparconv=1e-10, tolparinv=1e-10,maxiter=1000,geterrors=F)
str(samout27)


samout29<-SAMM(Y=matrix(ytrain,ncol=1),X=Xtrain,
Zlist=list(Ztrain, Ztrain), Klist=list(K1,K2),
lambda=0, W=Wtrain,R=list("ar1",c(0),matrix(5,1,1)),
Siglist=list("","",""), corfunc=c(F,F,T),
corfuncfixed=c(F,F,F), sigfunc=c(F,F,F),
mmalg="dermm_reml1", tolparconv=1e-10,
tolparinv=1e-10,maxiter=1000,geterrors=F)

str(samout29)

samout30<-SAMM(Y=matrix(ytrain,ncol=1),X=Xtrain,
Zlist=list(Ztrain, Ztrain), Klist=list(K1,K2),
lambda=0, W=Wtrain,R=list("ar1",c(0),matrix(5,1,1)),
Siglist=list("","",""), corfunc=c(F,F,T), corfuncfixed=c(F,F,F),
sigfunc=c(F,F,F),mmalg="dermm_reml2", tolparconv=1e-10,
tolparinv=1e-10,maxiter=1000,geterrors=F)
str(samout30)


############################

n=100
M1<-matrix(rbinom(n*150, 2, .2)-1, nrow=n)
K1<-relmatcov_cppforR(c(0), M1)
M2<-matrix(rbinom(n*150, 2, .2)-1, nrow=n)
K2<-relmatcov_cppforR(c(0), M2)
M3<-matrix(rbinom(n*100, 2, .2)-1, nrow=n)
K3<-relmatcov_cppforR(c(0), M3)
M4<-matrix(rbinom(n*100, 2, .2)-1, nrow=n)
K4<-relmatcov_cppforR(c(0), M4)


covY<-2*K1+3*K2+1*K3+2*K4+3*diag(n)

Y<-10+crossprod(chol(covY),rnorm(n))


#training set
Trainset<-sample(1:n,80)
y=Y[Trainset]
X=matrix(rep(1, n)[Trainset], ncol=1)
```

```
Z=diag(n)[Trainset,]

X=X
y=y

samout35<-SAMM(Y=matrix(y,ncol=1),X=X,Zlist=list(Z,Z,Z,Z),
lambda=0,Klist=list(K1,K2,K3,K4), W=Z,R=list(diag(n)),
Siglist=list("","","","",""), corfunc=c(F,F,F,F,F),
corfuncfixed=c(T,T,T,T,T),sigfunc=c(F,F,F,F,F),
mmalg="mmmk_ml", tolparconv=1e-10,
tolparinv=1e-10,maxiter=1000,geterrors=F)
str(samout35)

samout36<-SAMM(Y=matrix(y,ncol=1),X=X,Zlist=list(Z,Z,Z,Z),
lambda=0.99999,Klist=list(K1,K2,K3,K4), W=Z,
R=list(diag(n)),Siglist=list("","","","",""),
corfunc=c(F,F,F,F,F), corfuncfixed=c(T,T,T,T,T),
sigfunc=c(F,F,F,F,F),mmalg="mmmk_ml", tolparconv=1e-10,
tolparinv=1e-10,maxiter=1000,geterrors=F)
str(samout36)


outmat<-c()
for (lambda in seq(0,.999999, length=30)){


  samout37<-SAMM(Y=matrix(y,ncol=1),X=X,
  Zlist=list(Z,Z,Z,Z),lambda=lambda,Klist=list(K1,K2,K3,K4),
  W=Z,R=list(diag(n)),Siglist=list("","","","",""),
  corfunc=c(F,F,F,F,F), corfuncfixed=c(T,T,T,T,T),
  sigfunc=c(F,F,F,F,F),mmalg="mmmk_ml",
  tolparconv=1e-10, tolparinv=1e-10,
  maxiter=1000,geterrors=F)


  outmat<-cbind(outmat,c(samout37$Vu*samout37$weights, samout37$Ve))

}
str(samout37)
colnames(outmat)<-seq(0,.999999, length=30)

maxmat<-max(c(outmat))
minmat<-min(c(outmat))

plot(seq(0,.999999, length=30),outmat[1,],
ylim=c(minmat-1, maxmat+1), col=2, type="l")
for (i in 2:5){
  par(new=T)
  plot(seq(0,.999999, length=30), outmat[i,],
  axes=F, ylim=c(minmat-1, maxmat+1), col=i+1, type="l", xlab="", ylab="")
}
```

```
## End(Not run)
```

`sigcovfuncs_cppforR` *Covariance and Sigma Functions*

Description

The "kernel" functions end with "cov_cppforR" and sigma functions end with "Sig_cppforR". Check table below and the examples for details and usage. Documentation for some of these functions is missing. Let K be a given covariance matrix, D be a given Euclidean distance matrix, let q be the dimension of desired sigma or covariance matrices.

Arguments

`params`	a numeric vector for the parameters (a mapping of the original parameters)
`data`	a numeric matrix (see examples and details)

Details

function name	function ref	params	data	formula
IdentSig_cppforR	Ident	1	matrix(q,1,1)	$A = \sigma I_q$
ar1hetcov_cppforR	ar1het	q	matrix(q,1,1)	$A_{ij} = \sigma_i\sigma_j\rho^{\|i-j\|}$, $\sigma_1 = 1$
arma11cov_cppforR	arma11	2	matrix(q,1,1)	$A_{ij} = (\lambda\rho^{\|i-j\|-1} * 1(i \neq j) + 1(i = j))$
FA1hetSig_cppforR	FA1het	2q-2	matrix(q,1,1)	$f_i f_j + \sigma_{ii}1(i = j)$ $f_1 = 1, \sigma_1 = 1$
FA1homSig_cppforR	FA1hom	q	matrix(q,1,1)	$f_i f_j + \sigma 1(i = j)$ $f_1 = 1$
compsymmcov_cppforR	compsymm	1	matrix(q,1,1)	$A_{ij} = \rho, i \neq j$; $A_{ij} = 1, i = j$
compsymmhetSig_cppforR	compsymmhet	q+1	matrix(q,1,1)	$A_{ij} = \sigma_i\sigma_j\rho, i \neq j$ $A_{ij} = \sigma_i\sigma_j, i = j$
compsymmhetcov_cppforR	compsymmhet	q	matrix(q,1,1)	$A_{ij} = \sigma_i\sigma_j\rho, i \neq j$ $A_{ij} = \sigma_i\sigma_j, i = j$ $\sigma_1 = 1$
compsymmhomSig_cppforR	compsymmhom	2	matrix(q,1,1)	$A_{ij} = \sigma\rho, i \neq j$; $A_{ij} = \sigma, i = j$
diagSig_cppforR	diag	q	matrix(q,1,1)	$diag(\sigma_1, .., \sigma_q)$
diagcov_cppforR	Diag	q-1	matrix(q,1,1)	$diag(1, \sigma_2, .., \sigma_q)$
expcov_cppforR	exp	1	M	$exp(-\sigma * d_{ij})$ M defines D
lincombcov_cppforR	lincomb	k	$(K_1; \ldots; K_k)$	$\sum_{j=1}^{k} w_j K_j$
rbfcov_cppforR	rbf	1	M	$exp(-\sigma * d_{ij}^2)$

				M defines D
ar1cov_cppforR	ar1	1	matrix(nrow, 1,1)	$A_{ij} = \rho^{\|i-j\|}$
relmatcov_cppforR	RelMat	1	M	Genetic Similarity+
			M is coded as -1, 0, 1	σI
unstrcov_cppforR	unstr	$\frac{q(q+1)}{2} - 1$	matrix(q,1,1)	$A_{ij} = \sigma_i \sigma_j \rho_{ij}$
				$\sigma_1 = 1$,
			$\rho_{ii} = 1$	
ConstMatcov_cppforR	Const	0	K	$A = K$
expdistcov_cppforR	expdist	1	D	$exp(-\sigma d_{ij})$
rbfdistcov_cppforR	rbfdist	1	D	$exp(-\sigma d_{ij}^2)$
splincov_cppforR	splin	1	D	$1 - \rho d_{ij}, \rho d_{ij} \leq 1$
				$0, \rho d_{ij} > 1$
sppowcov_cppforR	sppow	1	D	$\rho^{d_{ij}}$

Value

A kernel or sigma matrix.

Author(s)

Deniz Akdemir

Examples

```
## Not run:
library(SAMM)
n=100
nsample=80
rhotrans=5
ar1cov_cppforR(c(rhotrans),matrix(5))
rho=(2/pi)*atan(rhotrans)
rho
tan((pi/2)*(rho))

M1<-matrix(rbinom(n*300, 2, .2)-1, nrow=n)
K1<-relmatcov_cppforR(c(.01), M1)

M2<-matrix(rbinom(n*300, 2, .2)-1, nrow=n)
K2<-relmatcov_cppforR(c(0.03), M2)
W=(diag(5)[sample(1:5,n, replace=TRUE),])
covY<-3*K1+5*K2+10*(W%*%ar1cov_cppforR(c(rhotrans),matrix(5))%*%t(W))
K1[1:5,1:5]
dim(W)
dim(ar1cov_cppforR(c(6),matrix(5)))
Y<-10+crossprod(chol(covY),rnorm(n))


#training set
```

```
Trainset<-sample(1:n,nsample)
ytrain=Y[Trainset]
Xtrain=matrix(rep(1, n)[Trainset], ncol=1)
Ztrain=diag(n)[Trainset,]
Wtrain=W[Trainset,]

samout<-SAMM(Y=matrix(ytrain,ncol=1),X=Xtrain,
Zlist=list(Ztrain, Ztrain), Klist=list(K1,K2),
lambda=0, W=Wtrain,R=list("ar1",c(0),matrix(5,1,1)),
Siglist=list("","",""), corfunc=c(F,F,T), corfuncfixed=c(F,F,F),
sigfunc=c(F,F,F),mmalg="dermm_reml2", tolparconv=1e-10,
tolparinv=1e-10,maxiter=1000,geterrors=F)
samout$corfuncparamslist[[3]]
rhohat=(2/pi)*atan(samout$corfuncparamslist[[3]])
rhohat
ar1cov_cppforR(c(samout$corfuncparamslist[[3]]),matrix(5,1,1))

## End(Not run)
```

Index