Package 'infinitefactor'

July 22, 2025

Type Package

Title Bayesian Infinite Factor Models

Version 1.0

Date 2020-03-30

Author Evan Poworoznek

Maintainer Evan Poworoznek <infinitefactorpackage@gmail.com>

Description Sampler and post-processing functions for semi-parametric Bayesian infinite factor models, motivated by the Multiplicative Gamma Shrinkage Prior of Bhattacharya and Dunson (2011) <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3419391/>. Contains component C++ functions for building samplers for linear and 2-way interaction factor models using the multiplicative gamma and Dirichlet-Laplace shrinkage priors. The package also contains post processing functions to return matrices that display rotational ambiguity to identifiability through successive application of orthogonalization procedures and resolution of column label and sign switching. This package was developed with the support of the National Institute of Environmental Health Sciences grant 1R01ES028804-01.

License GPL-2

Imports Rcpp (>= 1.0.2)

Depends reshape2, ggplot2, stats, utils

LinkingTo Rcpp, RcppArmadillo

NeedsCompilation yes

Repository CRAN

Date/Publication 2020-04-03 13:00:02 UTC

Contents

`infinitefactor-package`

Bayesian Infinite Factor Models

Description

Sampler and post-processing functions for semi-parametric Bayesian infinite factor models, motivated by the Multiplicative Gamma Shrinkage Prior of Bhattacharya and Dunson (2011) <https://www.ncbi.nlm.nih.gov/pmc/ Contains component C++ functions for building samplers for linear and 2-way interaction factor models using the multiplicative gamma and Dirichlet-Laplace shrinkage priors. The package also contains post processing functions to return matrices that display rotational ambiguity to identifiability through successive application of orthogonalization procedures and resolution of column label and sign switching. This package was developed with the support of the National Institute of Environmental Health Sciences grant 1R01ES028804-01.

Details

The DESCRIPTION file:

Package:	infinitefactor
Type:	Package
Title:	Bayesian Infinite Factor Models
Version:	1.0
Date:	2020-03-30
Author:	Evan Poworoznek
Maintainer:	Evan Poworoznek <infinitefactorpackage@gmail.com>
Description:	Sampler and post-processing functions for semi-parametric Bayesian infinite factor models, motivated by the M
License:	GPL-2
Imports:	Rcpp (>= 1.0.2)
Depends:	reshape2, ggplot2, stats, utils
LinkingTo:	Rcpp, RcppArmadillo

Index of help topics:

```
amean                   Average over the third index of an array
del_mg                  Sampler Components
infinitefactor-package
                        Bayesian Infinite Factor Models
```

interactionDL	Factor regression model with interactions using the Dirichlet-Laplace shrinkage prior
interactionMGSP	Factor regression model with interactions using the Multiplicative Gamma Shrinkage Prior
jointRot	Resolve rotational ambiguity in samples of factor loadings and factors jointly
linearDL	Sample Bayesian linear infinite factor models with the Dirichlet-Laplace prior
linearMGSP	Sample Bayesian linear infinite factor models with the Multiplicative Gamma Shrinkage Prior
lmean	Average elements of a list
msf	Resolve label and sign switching in random matrix samples
plotmat	Plot a matrix
summat	Summarise a matrix from posterior samples

Perform sampling with the linearMGSP() and linearDL() functions for linear factor models, or interactionMGSP() and interactionDL() functions for factor regression models including 2-way interactions. See jointRot() or msf() for postprocessing.

Author(s)

Evan Poworoznek

Maintainer: Evan Poworoznek <infinitefactorpackage@gmail.com>

References

Bhattacharya, Anirban, and David B. Dunson. "Sparse Bayesian infinite factor models." Biometrika (2011): 291-306.

Bhattacharya, Anirban, et al. "Dirichlet-Laplace priors for optimal shrinkage." Journal of the American Statistical Association 110.512 (2015): 1479-1490.

Ferrari, Federico, and David B. Dunson. "Bayesian Factor Analysis for Inference on Interactions." arXiv preprint arXiv:1904.11603 (2019).

Examples

```
k0 = 5
p = 20
n = 100

lambda = matrix(rnorm(p*k0, 0, 0.01), ncol = k0)
lambda[sample.int(p, 40, replace = TRUE) +
         p*(sample.int(k0, 40, replace = TRUE)-1)] = rnorm(40, 0, 1)
lambda[1:7, 1] = rnorm(7, 2, 0.5)
lambda[8:14, 2] = rnorm(7, -2, 0.5)
lambda[15:20, 3] = rnorm(6, 2, 0.5)
lambda[,4] = rnorm(p, 0, 0.5)
lambda[,5] = rnorm(p, 0, 0.5)
plotmat(varimax(lambda)[[1]])
```

```
X = matrix(rnorm(n*k0),n,k0)%*%t(lambda) + matrix(rnorm(n*p), n, p)

out = linearMGSP(X = X, nrun = 1000, burn = 500, adapt = FALSE)

aligned = jointRot(out$lambdaSamps, out$etaSamps)

plotmat(lmean(aligned$lambda))
```

---

amean *Average over the third index of an array*

Description

Convenience function to compute matrix sample means when samples are stored as a 3rd order array. Sampling index should be the third mode.

Usage

```
amean(ar)
```

Arguments

ar	a 3rd order array

Value

matrix of dimension dim(ar)[-3]

Author(s)

Evan Poworoznek

See Also

lmean

Examples

```
ar = array(rnorm(10000), dim = c(10, 10, 100))
amean(ar)
```

`interactionDL` *Factor regression model with interactions using the Dirichlet-Laplace shrinkage prior*

Description

Perform a regression of y onto X and all 2 way interactions in X using the latent factor model introduced in Ferrari and Dunson (2020). This version uses the Dirichlet-Laplace shrinkage prior as in the original paper.

Usage

```
interactionDL(y, X, nrun, burn = 0, thin = 1,
              delta_rw = 0.0526749, a = 1/2, k = NULL,
              output = c("covMean", "covSamples", "factSamples",
              "sigSamples", "coefSamples","errSamples"),
              verbose = TRUE, dump = FALSE, filename = "samps.Rds",
              buffer = 10000, adapt = "burn", augment = NULL)
```

Arguments

`y`	response vector.
`X`	predictor matrix (n x p).
`nrun`	number of iterations.
`burn`	burn-in period.
`thin`	thinning interval.
`delta_rw`	metropolis-hastings proposal variance.
`a`	shrinkage hyperparameter.
`k`	number of factors.
`output`	output type, a vector including some of: c("covMean", "covSamples", "factSamples", "sigSamples", "coefSamples", "numFactors", "errSamples").
`verbose`	logical. Show progress bar?
`dump`	logical. Save samples to a file during sampling?
`filename`	if dump: filename to address list of posterior samples
`buffer`	if dump: how often to save samples
`adapt`	logical or "burn". Adapt proposal variance in metropolis hastings step? if "burn", will adapt during burn in and not after.
`augment`	additional sampling steps as an expression

Value

some of:

covMean	X covariance posterior mean
omegaSamps	X covariance posterior samples
lambdaSamps	Posterior factor loadings samples (rotationally ambiguous)
etaSamps	Posterior factor samples (rotationally ambiguous)
sigmaSamps	Posterior marginal variance samples (see notation in Bhattacharya and Dunson (2011))
phiSamps	Posterior main effect coefficient samples in factor form (rotationally ambiguous)
PsiSamps	Posterior interaction effect coefficient samples in factor form (rotationally ambiguous)
interceptSamps	Posterior induced intercept samples
mainEffectSamps	Posterior induced main effect coefficient samples
interactionSamps	Posterior induced interaction coefficient samples
ssySamps	Posterior irreducible error samples

Author(s)

Evan Poworoznek

Federico Ferrari

References

Ferrari, Federico, and David B. Dunson. "Bayesian Factor Analysis for Inference on Interactions." arXiv preprint arXiv:1904.11603 (2019).

See Also

interactionMGSP

Examples

```
k0 = 5
p = 20
n = 50

lambda = matrix(rnorm(p*k0, 0, 0.01), ncol = k0)
lambda[sample.int(p, 40, replace = TRUE) +
         p*(sample.int(k0, 40, replace = TRUE)-1)] = rnorm(40, 0, 1)
lambda[1:7, 1] = rnorm(7, 2, 0.5)
lambda[8:14, 2] = rnorm(7, -2, 0.5)
lambda[15:20, 3] = rnorm(6, 2, 0.5)
lambda[,4] = rnorm(p, 0, 0.5)
lambda[,5] = rnorm(p, 0, 0.5)
```

```
plotmat(varimax(lambda)[[1]])

X = matrix(rnorm(n*k0),n,k0)%*%t(lambda) + matrix(rnorm(n*p), n, p)

beta_true = numeric(p); beta_true[c(1,3,6,8,10,11)] =c(1,1,0.5,-1,-2,-0.5)
Omega_true = matrix(0,p,p)
Omega_true[1,2] = 1; Omega_true[5,2] = -1; Omega_true[10,8] = 1;
Omega_true[11,5] = -2; Omega_true[1,1] = 0.5;
Omega_true[2,3] = 0.5;
Omega_true = Omega_true + t(Omega_true)
y = X%*%beta_true + diag(X%*%Omega_true%*%t(X)) +  rnorm(n,0.5)

intdl = interactionDL(y, X, 1000, 500, k = 5)
```

`interactionMGSP` *Factor regression model with interactions using the Multiplicative Gamma Shrinkage Prior*

Description

Perform a regression of y onto X and all 2 way interactions in X using the latent factor model introduced in Ferrari and Dunson (2020). This version uses the Multiplicative Gamma Shrinkage Prior introduced in Bhattacharya and Dunson (2011).

Usage

```
interactionMGSP(y, X, nrun, burn, thin = 1,
              delta_rw = 0.0526749, a = 1/2, k = NULL,
              output = c("covMean", "covSamples", "factSamples",
              "sigSamples", "coefSamples","errSamples"),
              verbose = TRUE, dump = FALSE, filename = "samps.Rds",
              buffer = 10000, adapt = "burn", augment = NULL)
```

Arguments

`y`	response vector.
`X`	predictor matrix (n x p).
`nrun`	number of iterations.
`burn`	burn-in period.
`thin`	thinning interval.
`delta_rw`	metropolis-hastings proposal variance.
`a`	shrinkage hyperparameter.
`k`	number of factors.
`output`	output type, a vector including some of: c("covMean", "covSamples", "fact-Samples", "sigSamples", "coefSamples", "numFactors", "errSamples").

verbose	logical. Show progress bar?
dump	logical. Save samples to a file during sampling?
filename	if dump: filename to address list of posterior samples
buffer	if dump: how often to save samples
adapt	logical or "burn". Adapt proposal variance in metropolis hastings step? if "burn", will adapt during burn in and not after.
augment	additional sampling steps as an expression

Value

some of:

covMean	X covariance posterior mean
omegaSamps	X covariance posterior samples
lambdaSamps	Posterior factor loadings samples (rotationally ambiguous)
etaSamps	Posterior factor samples (rotationally ambiguous)
sigmaSamps	Posterior marginal variance samples (see notation in Bhattacharya and Dunson (2011))
phiSamps	Posterior main effect coefficient samples in factor form (rotationally ambiguous)
PsiSamps	Posterior interaction effect coefficient samples in factor form (rotationally ambiguous)
interceptSamps	Posterior induced intercept samples
mainEffectSamps	Posterior induced main effect coefficient samples
interactionSamps	Posterior induced interaction coefficient samples
ssySamps	Posterior irreducible error samples

Author(s)

Evan Poworoznek

Federico Ferrari

References

Ferrari, Federico, and David B. Dunson. "Bayesian Factor Analysis for Inference on Interactions." arXiv preprint arXiv:1904.11603 (2019).

Bhattacharya, Anirban, and David B. Dunson. "Sparse Bayesian infinite factor models." Biometrika (2011): 291-306.

See Also

interactionMGSP

Examples

```
k0 = 5
p = 20
n = 50

lambda = matrix(rnorm(p*k0, 0, 0.01), ncol = k0)
lambda[sample.int(p, 40, replace = TRUE) +
         p*(sample.int(k0, 40, replace = TRUE)-1)] = rnorm(40, 0, 1)
lambda[1:7, 1] = rnorm(7, 2, 0.5)
lambda[8:14, 2] = rnorm(7, -2, 0.5)
lambda[15:20, 3] = rnorm(6, 2, 0.5)
lambda[,4] = rnorm(p, 0, 0.5)
lambda[,5] = rnorm(p, 0, 0.5)
plotmat(varimax(lambda)[[1]])

X = matrix(rnorm(n*k0),n,k0)%*%t(lambda) + matrix(rnorm(n*p), n, p)

beta_true = numeric(p); beta_true[c(1,3,6,8,10,11)] =c(1,1,0.5,-1,-2,-0.5)
Omega_true = matrix(0,p,p)
Omega_true[1,2] = 1; Omega_true[5,2] = -1; Omega_true[10,8] = 1;
Omega_true[11,5] = -2; Omega_true[1,1] = 0.5;
Omega_true[2,3] = 0.5;
Omega_true = Omega_true + t(Omega_true)
y = X%*%beta_true + diag(X%*%Omega_true%*%t(X)) +  rnorm(n,0.5)

intmgsp = interactionMGSP(y, X, 1000, 500, k = 5)
```

`jointRot` *Resolve rotational ambiguity in samples of factor loadings and factors jointly*

Description

Performs the varimax rotation on the factor loadings samples and column-based matching to resolve resultant sign and label switching. Rotates the factors along with the loadings to induce identifiability jointly. Note this method will only work on lists of factors and factor loadings that share the same constant number of factors (k) across all samples, and will likely crash the session if this is not the case.

Usage

```
jointRot(lambda, eta)
```

Arguments

`lambda`	list of factor loadings samples
`eta`	list of factor samples

Value

`lambda`	rotationally aligned factor loadings samples
`eta`	rotationally aligned factor samples

Author(s)

Evan Poworoznek

References

coming soon...

See Also

`msf`

Examples

```
k0 = 5
p = 20
n = 100

lambda = matrix(rnorm(p*k0, 0, 0.01), ncol = k0)
lambda[sample.int(p, 40, replace = TRUE) +
         p*(sample.int(k0, 40, replace = TRUE)-1)] = rnorm(40, 0, 1)
lambda[1:7, 1] = rnorm(7, 2, 0.5)
lambda[8:14, 2] = rnorm(7, -2, 0.5)
lambda[15:20, 3] = rnorm(6, 2, 0.5)
lambda[,4] = rnorm(p, 0, 0.5)
lambda[,5] = rnorm(p, 0, 0.5)
plotmat(varimax(lambda)[[1]])

X = matrix(rnorm(n*k0),n,k0)%*%t(lambda) + matrix(rnorm(n*p), n, p)

out = linearMGSP(X = X, nrun = 1000, burn = 500, adapt = FALSE)

aligned = jointRot(out$lambdaSamps, out$etaSamps)

plotmat(lmean(aligned$lambda))
```

`linearDL` *Sample Bayesian linear infinite factor models with the Dirichlet-Laplace prior*

Description

Perform Bayesian factor analysis by sampling the posterior distribution of parameters in a factor model with the Dirichlet-Laplace shrinkage prior of Bhattacharya et al.

Usage

```
linearDL(X, nrun, burn, thin = 1, prop = 1,
epsilon = 1e-3, k = NULL,
output = c("covMean", "covSamples", "factSamples",
"sigSamples"), verbose = TRUE, dump = FALSE,
filename = "samps.Rds", buffer = 10000,
augment = NULL)
```

Arguments

X	Data matrix (n x p)
nrun	number of iterations
burn	burn-in period
thin	thinning interval
prop	proportion of elements in each column less than epsilon in magnitude cutoff
epsilon	tolerance
k	Number of factors
output	output type, a vector including some of: c("covMean", "covSamples", "factSamples", "sigSamples")
verbose	logical. Show progress bar?
dump	logical. Save output object during sampling?
filename	if dump, filename for output
buffer	if dump, frequency of saving
augment	additional sampling steps as an expression

Value

some of:

covMean	X covariance posterior mean
omegaSamps	X covariance posterior samples
lambdaSamps	Posterior factor loadings samples (rotationally ambiguous)
etaSamps	Posterior factor samples (rotationally ambiguous)
sigmaSamps	Posterior marginal variance samples (see notation in Bhattacharya and Dunson (2011))
numFacts	Number of factors for each iteration

Author(s)

Evan Poworoznek

References

Bhattacharya, Anirban, et al. "Dirichlet-Laplace priors for optimal shrinkage." Journal of the American Statistical Association 110.512 (2015): 1479-1490.

See Also

linearDL

Examples

```
k0 = 5
p = 20
n = 50

lambda = matrix(rnorm(p*k0, 0, 0.01), ncol = k0)
lambda[sample.int(p, 40, replace = TRUE) +
         p*(sample.int(k0, 40, replace = TRUE)-1)] = rnorm(40, 0, 1)
lambda[1:7, 1] = rnorm(7, 2, 0.5)
lambda[8:14, 2] = rnorm(7, -2, 0.5)
lambda[15:20, 3] = rnorm(6, 2, 0.5)
lambda[,4] = rnorm(p, 0, 0.5)
lambda[,5] = rnorm(p, 0, 0.5)
plotmat(varimax(lambda)[[1]])

X = matrix(rnorm(n*k0),n,k0)%*%t(lambda) + matrix(rnorm(n*p), n, p)

out = linearMGSP(X = X, nrun = 1000, burn = 500)
```

---

linearMGSP *Sample Bayesian linear infinite factor models with the Multiplicative Gamma Shrinkage Prior*

Description

Perform Bayesian factor analysis by sampling the posterior distribution of parameters in a factor model with the Multiplicative Gamma Shrinkage Prior of Bhattacharya and Dunson

Usage

```
linearMGSP(X, nrun, burn, thin = 1, prop = 1,
epsilon = 1e-3, kinit = NULL, adapt = TRUE,
output = c("covMean", "covSamples", "factSamples",
"sigSamples", "numFactors"), verbose = TRUE,
dump = FALSE, filename = "samps.Rds", buffer = 10000,
augment = NULL)
```

Arguments

X	Data matrix (n x p)
nrun	number of iterations
burn	burn-in period
thin	thinning interval

prop	proportion of elements in each column less than epsilon in magnitude cutoff
epsilon	tolerance
kinit	initial value for the number of factors
adapt	logical. Whether or not to adapt number of factors across sampling
output	output type, a vector including some of: c("covMean", "covSamples", "factSamples", "sigSamples", "numFactors")
verbose	logical. Show progress bar?
dump	logical. Save output object during sampling?
filename	if dump, filename for output
buffer	if dump, frequency of saving
augment	additional sampling steps as an expression

Value

some of:

covMean	X covariance posterior mean
omegaSamps	X covariance posterior samples
lambdaSamps	Posterior factor loadings samples (rotationally ambiguous)
etaSamps	Posterior factor samples (rotationally ambiguous)
sigmaSamps	Posterior marginal variance samples (see notation in Bhattacharya and Dunson (2011))
numFacts	Number of factors for each iteration

Author(s)

Evan Poworoznek

References

Bhattacharya, Anirban, and David B. Dunson. "Sparse Bayesian infinite factor models." Biometrika (2011): 291-306.

See Also

linearDL

Examples

```
k0 = 5
p = 20
n = 50

lambda = matrix(rnorm(p*k0, 0, 0.01), ncol = k0)
lambda[sample.int(p, 40, replace = TRUE) +
         p*(sample.int(k0, 40, replace = TRUE)-1)] = rnorm(40, 0, 1)
```

```
lambda[1:7, 1] = rnorm(7, 2, 0.5)
lambda[8:14, 2] = rnorm(7, -2, 0.5)
lambda[15:20, 3] = rnorm(6, 2, 0.5)
lambda[,4] = rnorm(p, 0, 0.5)
lambda[,5] = rnorm(p, 0, 0.5)
plotmat(varimax(lambda)[[1]])

X = matrix(rnorm(n*k0),n,k0)%*%t(lambda) + matrix(rnorm(n*p), n, p)

out = linearMGSP(X = X, nrun = 1000, burn = 500)
```

`lmean` *Average elements of a list*

Description

Convenience function to compute sample means when samples are stored as a list. List elements should be compatible with addition and scalar division (e.g. must share the same dimensions).

Usage

```
lmean(list)
```

Arguments

`list`	a list of parameter samples

Value

same type as a single element of the input list

Author(s)

Evan Poworoznek

See Also

`amean`

Examples

```
l = replicate(100, rnorm(10), simplify = FALSE)
lmean(l)
```

`msf` *Resolve label and sign switching in random matrix samples*

Description

The msf() function performs column-based matching of a matrix to a pivot to resolve rotational ambiguity remaining after the application of an orthogonalisation procedure on a list of Bayesian matrix samples. The msfOUT() and aplr() functions perform this same matching but instead of returning aligned samples as does msf(), msfOUT outputs the list of permutations and sign switches needed for alignment and aplr outputs a list of matrices permuted and re-signed by msfOUT() output. msfOUT() and aplr() are used in jointRot(). These functions are written in C++ and may crash the R session if passed inappropriate input.

Usage

```
msf(lambda, pivot)

msfOUT(lambda, pivot)

aplr(matr, perm)
```

Arguments

`lambda`	matrix to be aligned, named for a factor loadings matrix as in the Bhattacharya and Dunson 2011 notation
`pivot`	matrix to align with which to align lambda
`matr`	a matrix to apply permutations to
`perm`	a (possibly signed) permutation order for the matr matrix

Details

see the examples for suggested usage of msf and jointRot() for suggested usage of msfOUT() and aplr().

Author(s)

Evan Poworoznek

See Also

`jointRot`

Examples

```
lambda = diag(10)[,sample(10)] + 0.001
pivot = diag(10)
msf(lambda, pivot)

# fast implementation for a list of samples
k0 = 5
p = 20
n = 100

lambda = matrix(rnorm(p*k0, 0, 0.01), ncol = k0)
lambda[sample.int(p, 40, replace = TRUE) +
         p*(sample.int(k0, 40, replace = TRUE)-1)] = rnorm(40, 0, 1)
lambda[1:7, 1] = rnorm(7, 2, 0.5)
lambda[8:14, 2] = rnorm(7, -2, 0.5)
lambda[15:20, 3] = rnorm(6, 2, 0.5)
lambda[,4] = rnorm(p, 0, 0.5)
lambda[,5] = rnorm(p, 0, 0.5)
plotmat(varimax(lambda)[[1]])

X = matrix(rnorm(n*k0),n,k0)%*%t(lambda) + matrix(rnorm(n*p), n, p)

out = linearMGSP(X = X, nrun = 1000, burn = 500, adapt = FALSE)

vari = lapply(out$lambdaSamps, varimax)
loads = lapply(vari, `[[`, 1)

norms = sapply(loads, norm, "2")
pivot = loads[order(norms)][[250]]

aligned = lapply(loads, msf, pivot)
plotmat(summat(aligned))
```

`plotmat` *Plot a matrix*

Description

Plot an image of a matrix using ggplot2

Usage

```
plotmat(mat, color = "green", title = NULL, args = NULL)
```

Arguments

`mat`	Matrix to plot
`color`	Color scheme: "green", "red", or "wes"
`title`	optional plot title
`args`	optional additional ggplot arguments

Value

sends image to active graphics device or outputs a ggplot object

Note

Uses reshape2::melt which may be aliased with reshape::melt

Author(s)

Evan Poworoznek

Examples

```
mat = diag(1:9 - 5)
plotmat(mat)
```

Sampler Components *Sampler Components*

Description

These are the component full conditional or Metropolis-Hastings updates coded in C++ used in the samplers in this package. The functions follow naming conventions based on their greek letter notation in their respective original papers, cited below, and the paper they come from. Here _mg refers to a component of the Multiplicative Gamma Shrinkage prior of Bhattacharya and Dunson 2011, _dl refers to a component of the Dirichlet-Laplace shrinkage prior of Bhattacharya et al., _lin refers to a component of a linear factor model as in Bhattacharya and Dunson 2011, and _int refers to a component of a factor model with 2-way interactions as in Ferrari and Dunson 2020.

Author(s)

Evan Poworoznek

References

Bhattacharya, Anirban, and David B. Dunson. "Sparse Bayesian infinite factor models." Biometrika (2011): 291-306.

Bhattacharya, Anirban, et al. "Dirichlet-Laplace priors for optimal shrinkage." Journal of the American Statistical Association 110.512 (2015): 1479-1490.

Ferrari, Federico, and David B. Dunson. "Bayesian Factor Analysis for Inference on Interactions." arXiv preprint arXiv:1904.11603 (2019).

summat *Summarise a matrix from posterior samples*

Description

Provide a summary matrix from a list of matrix-valued parameter samples, returning the mean value for each element with 0 not included in its quantile-based posterior credible interval, and 0 for each element for which 0 is included in its posterior CI.

Usage

```
summat(list, alpha = 0.05)
```

Arguments

list	list of matrix valued parameter samples of the same dimensions
alpha	type I error probability

Value

a matrix

Author(s)

Evan Poworoznek

See Also

lmean

Examples

```
list = replicate(1000, matrix(rnorm(100), ncol = 10) +
                 10*diag(10), simplify = FALSE)
lmean(list)
summat(list)
plotmat(summat(list))
```

Index