Package ‘restrictedMVN’

December 27, 2016

Type Package
Title Multivariate Normal Restricted by Affine Constraints
Version 1.0
Date 2016-12-14
Author Jonathan Taylor and Yuval Benjamini
Maintainer Yuval Benjamini <yuval.benjamini@mail.huji.ac.il>
Description A fast Gibbs sampler for multivariate normal with affine constraints.
License GPL (>= 2)
Imports MASS
Suggests testthat

R topics documented:

restrictedMVN-package .................................................. 2
factor_covariance ....................................................... 3
sample_from_constraints .............................................. 3
thresh2constraints ...................................................... 4
whiten_constraint ...................................................... 5

Index 6
restrictedMVN-package

Sampler from multivariate normal with affine constraints

Description

The package implements a fast Gibbs sampler for the multivariate normal with affine constraints. For the d-dimensional Z~Normal(mu, sigma), the linear_part matrix A in d x r, and offset vector b in 1 x r define a multivariate normal with affine constraints in \( \{Z | A^T Z \leq b\} \).

Details

Sampling is implemented in the main function, `sample_from_constraints`. It is parameterized by the parameters of the normal (mean_param and covariance), parameters of the restriction (linear_part and offset), and the number of samples ndraw. The user also needs to specify an initial point that satisfies the constraints. `thresh2constraints` is a helper function that translates coordinate-wise truncations into the affine form.

Author(s)

Jonathan Taylor and Yuval Benjamini.

Maintainer: Yuval Benjamini <yuval.benjamini@mail.huji.ac.il>

See Also

The package was originally part of the github selective-inference code base.

Examples

```r
constr = thresh2constraints(3, lower = c(0.2, 0.2, 0.2))
covariance = matrix(c(1, 0.5, 0.5, 0.5, 1, 0.5, 0.5, 0.5, 1), ncol=3)

samp = sample_from_constraints(linear_part = constr$linear_part,
                               offset = constr$offset,
                               mean_param = c(0, 0, 0),
                               covariance = covariance,
                               initial_point = c(1, 1, 1),
                               ndraw=20000,
                               burnin=2000)

# all points should be >= 0.2
stopifnot(all(samp>=0.2))

mean_restricted = colMeans(samp)

# compare to rejection of multivariate normals
library("MASS")
full_samp = mvrnorm(n=100000, mu = c(0, 0, 0), Sigma = covariance)
# Add restrictions:
```
pass_restrictions = apply(full_samp, 1,
    function(x)(all(constr$linear_part%*% x - constr$offset <= 0 )))

cond_samp = full_samp[pass_restrictions,]
mean_restricted_rej = colMeans(cond_samp)

stopifnot(all(abs(mean_restricted - mean_restricted_rej)<=0.05))

factor_covariance

Compute the square-root and inverse square-root of a non-negative definite matrix.

Description

Compute the square-root and inverse square-root of a non-negative definite matrix.

Usage

factor_covariance(S, rank = NA)

Arguments

S matrix
rank rank of svd

sample_from_constraints

Sample from multivariate normal distribution under affine restrictions

Description

sample_from_constraints returns a sample from the conditional multivariate normal, restricted by affine constraints. The constraints are coded by a linear matrix and an offset vector: linear_part %*% Z <= offset. The sampling uses a Gibbs sampler, and requires an initial vector that meets the restriction.

Usage

sample_from_constraints(linear_part, offset, mean_param, covariance,
                         initial_point, ndraw = 8000, burnin = 2000)
Arguments

linear_part  \( r \times d \) matrix for \( r \) restrictions and \( d \) dimension of \( Z \)
offset        \( r \)-dim vector of offsets
mean_param    \( d \)-dim mean vector of the unconditional normal
covariance    \( d \times d \) covariance matrix of unconditional normal
initial_point \( d \)-dim vector that initializes the sampler (must meet restrictions)
ndraw         size of sample
burnin        samples to throw away before storing

Value

\( Z \) \( n_{\text{draw}} \times d \) matrix of samples

Examples

```r
# Compute conditional mean of correlated lower-truncated vector
constr = thresh2constraints(3, lower = c(1,1,1))
covariance = matrix(c(1,0.5,0,0.5,1,0.5,0,0.5,1),nc=3)
samp = sample_from_constraints(linear_part = constr$linear_part,
                             offset= constr$offset,
                             mean_param = c(0,0,0),
                             covariance = covariance,
                             initial_point = c(1.5,1.5,1.5),
                             ndraw=500,
                             burnin=2000)

# all points should be >= 1
any(samp<1)
colMeans(samp)
```

thresh2constraints  \( \) Translate between coordinate thresholds and affine constraints

Description

thresh2constraints translates lower and upper constraints on coordinates into linear and offset constraints (\( A*Z \leq B \)). lower and upper can have -Inf or Inf coordinates.

Usage

thresh2constraints(d, lower = rep(-Inf, d), upper = rep(Inf, d))
whiten_constraint

Arguments

- **d**: dimension of vector
- **lower**: 1 or d-dim lower constraints
- **upper**: 1 or d-dim upper constraints

whiten_constraint  Transform non-iid problem into iid problem

Description

Transform non-iid problem into iid problem

Usage

whiten_constraint(linear_part, offset, mean_param, covariance)

Arguments

- **linear_part**: matrix, linear part of constraints
- **offset**: vector, bias of constraints
- **mean_param**: vector of unconditional means
- **covariance**: vector of unconditional covariance

Value

new linear_part and offset for 0-mean iid covariance problem, and functions that map between the two problems.
Index

*Topic package
  restrictedMVN-package, 2

factor_covariance, 3

restrictedMVN(restrictedMVN-package), 2
restrictedMVN-package, 2

sample_from_constraints, 3

thresh2constraints, 4

whiten_constraint, 5